An Intelligent System for Relational Databases

Uday Prakash Gunjal[1], Vaibhav Rathod[2], Dr. Nitin N. Pise[3]

[1, 2]ME Computer Engineering Student, Maharashtra Institute of Technology, Pune

[3]Assistant Professor, Computer Engineering Department, Maharashtra Institute of Technology, Pune

Abstract: ***Today's present computing world, most new-generation database applications demand intelligent interface to enhance efficient interactions between database and the users. The most accessible interfaces for databases must be intelligent and able to understand natural language expressions. In this paper mapping of natural language queries to SQL is suggested. We propose a general architecture for an intelligent database system and also a real implementation of such a system which can be connected to any database. One of the main characteristics of this interface is domain-independence, which means that this interface can be used with any database. Another characteristic of this system is ease of configuration. The intelligent system employs semantic matching technique to convert natural language query to SQL using dictionary and set of production rules. The dictionary consists of semantics sets for tables and columns. The shaped query is executed and the results are presented to the user. This interface was first tested using Supplier-Parts database and secondly with Northwind database of SQL server 7.0.***

Keywords: Databases, Structured Query Language (SQL), intelligent interface, Intelligent Database System (IDBS), Flexible Querying, Intelligent Layer, Domain Independent Interface

1. Introduction

In the present computing world, computer based information technologies have been extensively used to help many organizations, private companies, academic and education institutions to manage their processes and information systems. Information systems are used to manage data. A general information management system that is capable of managing several kinds of data, stored in the database is known as Database Management System (DBMS) [1]. Databases are comprehensive element in private and public information systems which are essential in number of application areas[2]. Databases are gaining prime importance in a huge variety of application areas employing private and public information systems. Retrieval of a large amount of the same type of data is very efficient in relational databases [3], but still the user has to master the DB schema completely to formulate the queries.

Structured Query Language (SQL) is an ANSI standard for accessing and manipulating the information stored in relational databases. It is comprehensively employed in industry and is supported by major database management systems (DBMS). Most of the languages used for manipulating relational database systems are based on the norms of SQL.

In the past few years, many advances have been made in the field of databases and many other fields of critical relevance to information technology. An intelligent database is an emerging database technology that has dramatic impact on the way we think and work [4]. It greatly expands our capabilities as information users. Intelligent databases are the databases that that are endowed with data management system able to manage large quantities of data to which various forms of reasoning can be applied to infer additional data and information. This includes knowledge representation techniques, inference techniques and intelligent user interfaces.

These techniques play different roles in enhancing database systems: knowledge representation techniques allow one to represent better in DB semantics of the application domain, inference techniques allow one to reason about the data to extract additional data or information; and intelligent user interfaces help users to make requests and receive the replies interfaces by making use , typically , of natural language (NL) facilities that extend beyond the traditional query language approach [5] provided. The formatter will need to create these components, incorporating the applicable criteria that follow.

2. Natural Language Interface to Database

In recent times, there has been a rising demand for non-expert users to query relational databases in a more natural language encompassing linguistic variables and terms, instead of operating on the values of the attributes. Talking to a computer in a natural language such as plain English is always a dream that drives the progress of human-computer interaction work [6,7].

This has led to the development of Natural Language Interface to Databases (NLIDB). NLIDBs permits users to formulate queries in natural language, providing access to information without requiring knowledge of programming or database query languages. There are many applications that can take advantages of NLIDB. In PDA and cell phone environments, the display screen is not as wide as a computer or a laptop. Filling a form that has many fields can be tedious: one may have to navigate through the screen, to scroll, to look up the scroll box values, etc. Instead, with NLIDB, the only work that needs to be done is to type the question similar to the SMS (Short Messaging System).

The NLIDB is actually a branch of more comprehensive method called Natural Language Processing (NLP). In general, the main objective of NLP research is to create an easy user friendly environment to interact with users in the sense that computer does not require any programming language skills to access the data; only natural language is required.

The research of Natural Language interface to databases (NLIDB) has recently received attention from the research communities. The area of NLIDB research is still very experimental and systems so far have been limited to small domains, where only certain types of statements can be used. When the systems are scaled up to cover larger domains, it becomes difficult due to vast amount of information that needs to be incorporated in order to parse statements. Although the earliest research has started since the late sixties, NLIDB remains an open search problem.

3. Related Work

For the last thirty years, numerous attempts have been made to build useful natural language interface. It has turned out to be much more difficult than what was originally expected. There had large number of research works introducing the theories and implementations of NLIDBs. There are mainly four kinds of NLIDBs framework.

A. Template Based approach
The first type of framework is based on pattern matching. Typical applications of this type of framework is SAVVY[. In this system, some patterns are written for different types of queries and these patterns are executed after the queries are entered. The main advantage of pattern matching approach is that no elaborate parsing and modules of interpretation are required and the systems are very easy to implement. Some pattern matching systems were able to perform impressively well in certain applications. One of the best natural language processing system that is based on pattern-matching approach is ELIZA. However a pattern matching system is too shallow and therefore would often lead to bad failures.

B. Syntax-based approach
The method used in the system supports a syntax-based approach where a parsing algorithm is used to generate a parse tree depending on user‟s queries. This method is especially used in application-specific database systems. A database query language must be provided by the system to enable the mapping from parse tree to the database query. Moreover, it is difficult to decide the mapping rules from the parse tree to the query language (e.g. SQL) that the database uses.

C. Semantic Grammar approach
System uses semantic grammars where syntactic processing techniques and semantic processing techniques are used together. The disadvantage of this method is that semantic approach needs a specific knowledge domain, and it is quite difficult to adapt the system to another domain. In fact, a new grammar has to be developed when the system is configured for a different domain.

D. Intermediate Representation language based approach
Some intermediate representation languages can be used to convert the statements in natural language to a known formal query language. MASQUE/SQL is an example for this approach. It is a front-end language for relational databases that can be reached through SQL. User defines the types of the domain which database refers using a hierarchy in a built-in domain-editor. Moreover, words expected to appear in queries with their logical predicates are also declared by the user. Queries are first transformed into a Prolog-like language LQL, then into SQL. The advantage of this technique is that the system generating the logic queries is independent from the database and therefore, it is very flexible in domain replacements.

Despite of the achievements attained in this area, present day NLIDBs do not guarantee correct translation of queries in natural language to database languages. The most desirable characteristic of the NLIDBs that the researchers are proposing is Domain Independence, which means that interface can be used with different databases and reconfiguration of the NLIDB from one domain to another is done automatically.

In these types of interfaces, the percentage of correctly answered queries ranges from 69.4% to 96%. Regarding the main domain independent interfaces the following stand out: EnglishQuery, PRECISE, ELF, Edite, SystemX, and MASQUE/SQL. The success percentage of these interfaces is less than that of domain dependent interfaces. Much of the research work is being done to improve the success percentage of domain independent interfaces.

4. The Intelligent System Overview

It is known that databases respond only to standard SQL queries and it is highly impossible for a common person to be well versed in SQL querying. Moreover they may be unaware of the database structures namely table formats, their fields with corresponding types, primary keys and more. On account of these we design 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 main characteristic of the system are as follows:
- It is domain independent. The Hence, the configuration process is automatic.
- The interface can be easily and automatically configured. Interface relies on the Metadata set, Semantic set for tables and attributes

The primary advantage of the system is that it conceals the inherent complexity involved in information retrieval based on unqualified user queries. Paper [7] introduced core algorithm for converting a natural language user query to Standard SQL query. That paper provides a brief overview of the domain independent NLIDB system focusing on the components necessary to understand the functionality of the proposed system.

In the presented system, an intelligent layer is designed in such a way that it can be can be connected to any existing database system, which is responsible for the intelligent information processing and performing flexible queries.

The system is designed to accept any relational database schema. 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. Proposed system is composed of two modules: **a pre-processor** and **a run time processor**. Pre-processor is used to generate Domain Dictionary. Dictionaries used by the existing NLIDBs are created manually or semi automatically 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, on WordNet and on the administrator feedback. The system administrator can edit, add and modify these rules.

The **run time processor** uses these rules and tries to match the input words with predefined data structures, tables and attribute names from the database schema. The rules describe the relations between the table and its attributes. 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.

5. Design of Intelligent Interface For Databases

In general, a database (*D*) 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*). Table I. shows the sample metadata set of the supplier-part database which is automatically generated by the system. The metadata set holds entries for all the „*n*" tables in the relational database with all their corresponding fields and their unique primary key

Table 1: Metadata Set

Table	Primary Key	Field	Foreign Key
Supplier	Sup_no	Sup_no, Sup_name, sup_city	
Part	Part_no	Part_no, Part_name, Part_city, Part_color	
Shipment	Part_no,Sup_no	Sup_no, Part_no, Quantity	(Sup_no, Supplier) (Part_no, Part)

The metadata set is organized as follows:
M = {T_y, F_{Ty}, P_{ky}, F_{ky}, R_{ky} }, $0 < y < (n+1)$
Where $T \rightarrow$Set of tables in *M*
$FT \rightarrow$Set of all fields in *T*
$P_K \rightarrow$Primary key in *T*
$F_K \rightarrow$Foreign key in *T*
$R_K \rightarrow$Reference attribute and table

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:

- The Expression mapping set (E_{map}) contains the list of commonly used conditional clauses and their associated mathematical symbols. It acts as a look up table to locate the SQL defined mathematical operators
- The Conjunction training set (C_T) consists of the list of generally used Conjunctive clauses like where, who 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.
- The semantic set (S) contains the list of all possible semantics related to table names and fields in the database.
- Rules related to database schema-The schema of the database gets translated into the rules. These rules are produced by the system, and they are based on the relationships between the tables. If in a SQL query two tables Suppliers and Products are referenced , then the attribute equation Products.CategoryId = Categories.CategoryId should be used in the where clause.

A. Intelligent Interface components

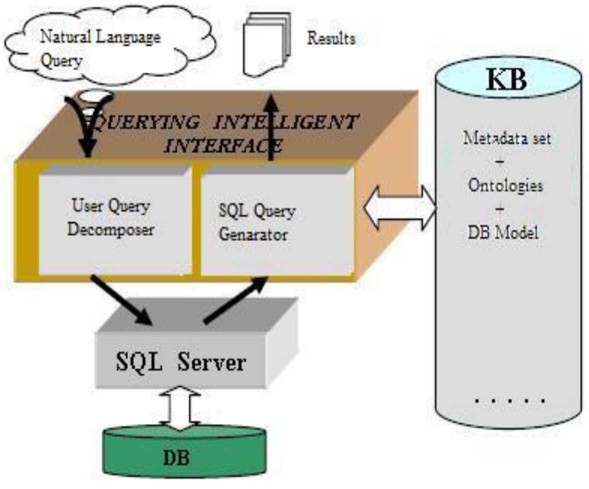

Figure 1: Components of intelligent interface

This section gives a vivid description of how the user query is transformed to be used for data retrieval from databases. In our proposed approach, we define a universal set Q_u which holds all the individual tokens in the user typed query. Every token represents a unique element in the universal set Q_u. Every user query is likely to contain a display or subjective part which specifies the intended result, the Conjunction part which determines the SQL definition Clause and the Criteria part which describes the condition or constraint. All these parts of the user query will be represented as three distinct sets P_d, P_j and P_c . Here the sets P_d, P_j and P_c contain tokens that represent the subject, the Conjunctive clause and the Criteria respectively.
$Q_u \rightarrow$ User Query
$P_d \rightarrow$ Subjective/Display part of Q_u
$P_j \rightarrow$ Conjuctive part of Q_u
$P_c \rightarrow$ Criteria/condition part of Q_u

Evidently, the sets P_d , P_j and P_c form the subsets of the universal setQ_u . The subsets of Q_u may be structured in the following formats:

$Q_u = \{P_d\}$ Or $Q_u = \{P_d , P_j , P_c \}$ Or
$Q_u = \{P_c , P_j , P_d\}$
The two components of intelligent interface as shown in Figure I are User query decomposer and SQL query generator, the functions of which are described as follows.

B. User Query Decomposer
This component decomposes the User Query into three parts P_d , P_j and P_c using the predefined training sets. C_T , E_{map} and S_W. Firstly user query is intersected with C_T. This decomposes the query in above three parts. Then expression mapping is done for Pc using E_{map}. Subsequently stop word removal is done for P_d and P_c using S_W training set.
Example 1:
Get me supplier details where supplier‟s
native place is LONDON
$Q_u \rightarrow$ Get me supplier details where supplier‟s
native place is LONDON
$P_d \rightarrow$ Supplier details
$P_j \rightarrow$ where
$P_c \rightarrow$Supplier native place = LONDON

Example 2:
Get full details of all the suppliers
$Q_u \rightarrow$ Get full details of all the suppliers
$P_d \rightarrow$ Details suppliers
$P_j \rightarrow$*null*
$P_c \rightarrow$ *null*
$P^t_d \rightarrow$Table (P_d) and consider $P^t_d = \Phi$(initially)
$P^f_d \rightarrow$Table (P_d) and consider $P^f_d = \Phi$(initially)

The next step is to locate the tables and fields mentioned in the set P_d and P_c . These sets are intersected with the metadata set M. If it yields a nonempty set, table names are chosen from the Metadata set M. Else if it yields an empty set, then the sets are intersected with the set S_T and S_F, to retrieve the appropriate table names and field names associated with the matching semantics.

Now P^t_d contains the names of tables to used in SQL query and P^f_d contains fields names and P_c contains condition part of the query.

C. SQL Query Generator
This component finds the relation between the tables display condition and the fields that are to be displayed and appends the joining condition of tables to P_c . With help of P^t_d , P^f_d and P_c user query in natural language is converted to equivalent SQL query.

If $P_c \neq \Phi$ then
SQL Query = SELECT P^f_d FROM P^t_d AS A WHERE P_c

Else
SQL Query = SELECT P^f_d FROM P^t_d
So the Generated SQL Statement for Example 1 is
SELECT * FROM Supplier AS A WHERE A.Sup_city=‟LONDON‟

6. Experimental Results

In this section, we have presented the experimental results of the proposed intelligent interface. The presented system has been implemented in JAVA with MySQL and MS-Access as databases.

For the experiment, supplier-part database and Northwind database of SQL server 7.0 were used and a group of five students was asked to formulate the queries in English for the two databases. The students were explained about the schema, relationships between the tables, primary keys etc. of the database for formulating the queries. Additionally, the students were given brief explanation about the kind of queries they could formulate. The resulting corpus consists of 20 queries for Supplier-Part database and 40 queries for the Northwind database. Most of the queries selected for the Northwind database involved two or more tables. The queries were classified according to difficulty and were divided into
four types :

(1) Explicit table and column information
(2) Explicit table and implicit column
(3) Implicit table and explicit column and
(4) Implicit table and column information.

The corpus of the queries for the supplier-part database and the Northwind database was introduced to the interface to obtain the percentage of queries correctly answered and queries with incorrect answer or queries unanswered. The following results were obtained : 75% of the queries (considering all the four types) of the supplier-part database and 70% of the queries of Northwind database were translated were translated correctly.

The main reason that caused these errors was due to insufficient information for processing. Some examples of the queries that could not be correctly translated are the following: *Get full details of LONDON suppliers?* And *Get the unit price of TOFU*. In both the queries the attributes are not specified. If the first query is rephrased as: *Get full details of suppliers whose native place is LONDON* or *Get full details of suppliers whose city is LONDON*, the interface translates it correctly because the attribute *city* is specified. Similarly, in the second query attribute of TOFU is not specified. If the second query is rephrased as : *Get the unit price of product whose name is TOFU* then the interface matches name with the attribute *product name* and the query is translated correctly.

7. Conclusion

Flexible intelligent system for database querying is presented in this paper. The main advantage of the system is natural language is used for querying the database and incorporated into the existing database systems. The presented system accepts flexible user queries and converts them into a standard SQL query. Expression mapping, stop words removal and semantic matching techniques have been utilized by the intelligent layer in the formation of the SQL query. The efficacy of the presented system has been demonstrated with the aid of experimental results.

References

[1] N. Sangeeth, R. Rejimoan, “An Intelligent System for Information Extraction From Relational Database Using HMM,” International Conference on Soft Computing Techniques and Implementations(ICSCTI). Dept. of ECE, Faridabad,India, Oct 2015

[2] N Nihalani, Sanjay Silakari, Mahesh Motwani , “Design of Intelligent layer for flexible querying in databases,” International Journal of Computer Science and Engineering (IJCSE) Vol 1(2) 2009

[3] Zongmin Ma, "Intelligent Databases: Technologies and Applications", IGI publishing, 320 pages, 2007K.

[4] B. Juan J. González, Rodolfo A. Pazos Rangel, I. Cristina Cruz C., H. Héctor J. Fraire and L. de Santos Aguilar, et al.(2006) "Issues in Translating from Natural Language to SQL in a Domain- Independent Natural Language Interface to Databases"LNCS 4293, MICAI: Advances in Artificial Intelligence, pp 922-931

[5] Dietmar Wolfram, "Applications of SQL for Informetric Data Processing", Proceedings of the 33rd conference of the Canadian Association for Information Science, 2005

[6] Donald P. Mckay and Timothy W. Finin, "The Intelligent Database Interface: Integrating AI and Database systems", In Proceedings of the 1990 National Conference on Artificial Intelligence, pp. 677-684, 1990

[7] N Nihalani, Sanjay Silakari, Mahesh Motwani “An Intelligent Interface for relational databases,” IJSSST, Vol. 11, No. 1, ISSN: 29 1473-804x online, 1473-8031