International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 05 Issue: 03 | Mar-2018 www.irjet.net p-ISSN: 2395-0072

An Efficient Way to Querying XML Database Using Natural Language

Prof. P. D. Thakare[1], Ms. Snehal Dhole[2]

[1]*Head of the Department, Dept. of Computer Science, JCOET Yavatmal, Maharashtra, India*
[2]*Student, Dept. of Computer Science, JCOET Yavatmal, Maharashtra, India*

Abstract - *In today's world databases are widely used in many fields like banking, human resources, universities, corporate, hotels, government organisations etc. Everyone needs to deal with databases for the extraction of required data. But everyone does not have knowledge of query processing languages such as XQUERY or SQL etc. So, it is very difficult for a common user to retrieve information from the database using database query languages. Access any kind of data from database using natural language like English is a convenient and easy method instead of using formal query languages such as XQuery, SQL etc. The proposed system can accept English language sentences and then it is translated into an XQuery/SQL expression. This XQuery/SQL statement can be evaluated against an XML database. This query translation is done by mapping the tokens in the dependency parse tree of the natural language query into the XQuery fragments. This paper introduces a novel architecture which can translate a wide range of natural language including arithmetic operations queries into formal database queries by achieving maximum accuracy.*

Key Words: XML database, XQuery, SQL, NLQ, XML, NLIDB, etc.

1. INTRODUCTION

In the todays real world, generally we used natural language, such as English to obtain the information. Not astonishingly, supporting arbitrary natural language queries is regarded by many as the ultimate goal for a database query interface, and there have been numerous attempts toward this goal. Nevertheless, two major obstacles lie in the way of reaching the ultimate goal of support for arbitrary natural language queries: first, automatically understanding natural language is itself still an open research problem, not just semantically but even syntactically; second, even if we could fully understand any arbitrary natural language query, translating this parsed natural language query into a correct formal query would remain an issue since this translation requires mapping the understanding of intent into a specific database schema.

We use Database as most commonly and broadly use application for storing and retrieving large quantity of data instantaneously and proficiently. However, retrieving data from these databases is not an easy job. To communicate with the databases the languages like XQuery, SQL etc. are used. Using natural language to interact with a database system becomes very important since the use of database systems is widespread and their accessibility to common users is needed to get the complete use of the database system. The idea of using natural language instead of formal query languages for retrieving information in a database is a latest application of NLP called Natural Language Interface to Databases(NLIDB). NLIDB is a step towards the development of Intelligent Databases Systems (IDBS) to allow the users to perform flexible querying in databases. The use of Natural Language I. Interfaces to Databases (NLIDB) is to allow the users to ask questions in natural language and receive responses. Like other systems NLIDB systems also have some advantages and disadvantages.

There are problems with this interface because we cannot predict what type of questions the NLIDB can work with ie. the linguistic coverage of the system is indefinite. And some linguistic operations which takes place inside the system can consume time for execution. The operations performed in a NLIDB system can be considered as two processing stages; Linguistic processing and Database processing. In the first stage, the components of natural language query (NLQ) is mapped and translated into the corresponding database query fragments. In the database pro- cessing stage, database management and access is performed, and the query is executed by the system.

In this paper, we propose a framework for building a generic interactive natural language interface to database systems. Our focus is on the second challenge: given a parsed natural language query, how to translate it into a correct structured query against the database. The issues we deal with include those of attribute name confusion (e.g., asked "Who is the president of YMCA?" we do not know whether YMCA is a country, a corporation, or a club) and of query structure confusion (e.g., the query "Return the lowest price for each book" is totally different from the query "Return the book with the lowest price," even though the words used in the two are almost the same). We address these issues in this article through the introduction of the notions of token attachment and token relationship in natural language parse trees. The input natural language query sentences are converted to XQuery statements. XQuery is a language for finding and extracting elements and attributes from XML documents. XQuery for XML is like SQL for databases [2], [3]

We also propose the concept of core token as an effective mechanism to perform semantic grouping and hence determine both query nesting and structural relationships between result elements when mapping tokens to queries.

2. EXISTING WORK

As many works have been done in the field 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 a logical representation of the sentence followed by a conversion of the representation into a database query.

Most current NLIDBs first transform the natural language question into an intermediate logical query, expressed in some internal meaning representation language. The intermediate logical query expresses the meaning of the user's question in terms of high level world concepts, which are independent of the database structure. The logical query is then translated to an expression in the database's query language, and evaluated against the database [9].

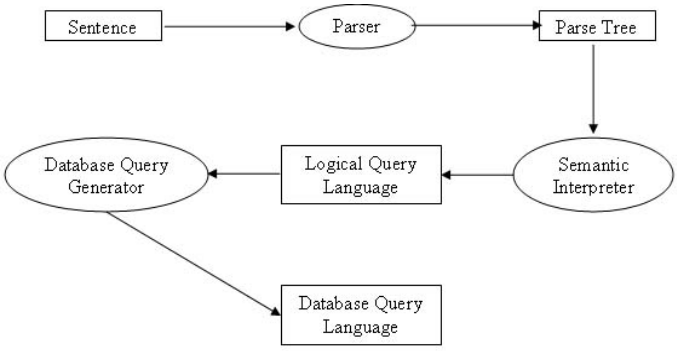

Fig-1 Intermediate Representation of Language

As figure shows intermediate representation language, the system can be divided into two parts[6]. One part starts from a sentence up to the generation of a logical query. The other part starts from a logical query until the generation of a database query. In the part one, The use of logic query languages makes it possible to add reasoning capabilities to the system by embedding the reasoning part inside a logic statement. In addition, because the logic query languages is independent from the database, it can be ported to different database query languages as well as to other domains, such as expert systems and operating systems.

Yunyao Li et. al. proposed a framework called NaLIX for building a natural language interface to XML data. The relationship between words in the NLQ is obtained by analyzing output from the MINIPAR dependency parser, which discovers the grammatical dependencies between tokens rather than hierarchical components[6]. The tokens in the input sentence are classified into different categories. The different parts of the output query are formed with the help of parse tree and tokens of the input. The XQuery fragments are joined together to construct the complete XQuery by determining the grouping and nesting of fragments. Another work of these authors is DaNaLIX which is a prototype domain-adaptive natural language interface for querying XML data. A domain adapter used in this system checks the available domain knowledge and modifies the dependency tree by applying some linguistic rules. Domain knowledge is represented as a collection of rules representing the mapping between a partial parse tree containing terms with domain meanings with one expressed in terms understandable by a system like NaLIX.

NLKBIDB is a combined approach of NLIDB and knowledge-based interface to database(KBIDB). NLIDB allows generating accurate SQL query by using the SQL generation rule based on natural language concept. And KBIDB handles the sentences which are grammatically correct or incorrect but SQL generation rules of KBIDB is weaker than that of NLIDB. For grammatically correct sentences, NLKBIDB applies the query construction rules of NLI. And for the sentences with grammar mistakes, it applies rules of KBI for SQL query construction.

The architecture proposed by Fei Li, and H. V. Jagdish is known as NaLIR. It comprises of three functional parts: a first component that transforms a NLQ to a query tree, a second component that verifies the transformation by communicating with the user, and a third component that translates the query tree into a SQL statement. The query interpretation part, which includes parse tree node mapper and structure adjustor, is responsible for interpreting the NLQ and representing the interpretation as a query tree. An interactive communicator is used to communicate with the user to ensure that the interpretation process is correct. The query tree, verified by the user, is then translated into a SQL statement in the query tree translator and then evaluated against an RDBMS.

In the system proposed by F.Siasar djahantighi et al. they introduced a method which prepares an expert system that can identify synonymous words in any language. It first parses the natural language sentences, and then the sentences are transformed to SQL queries. Preprocessor is used here to create a semantic database from different kinds of rules of the language and semantic sets for all entities and all their possible attributes. This database helps to return the same query for more than one sentence with the same meaning.

Mapping of the NLQ to SQL query is done by matching the input with some particular patterns.

Alessandra Giordani and Alessandro Moschitti approached the problem of converting natural into query language semantics by the automatic learning of a model from the lexical and syntactic description of the training data samples. And these are pairs of NL questions and SQL queries, which are expounded using a semi-supervised algorithm. Kernel methods and support vector machines are used in this system to represent syntactic or semantic relationships expressed by the pairs.

D-HIRD is an NLIDB system proposed by Rajender Kumal, Mohit Dul, and Shivani Jinda that accepts a NLQ in Hindi and then it generates corresponding SQL query, and returns the result by extracting the data from corresponding databases. The system uses the Hindi Shallow Parser and lexicon for linguistic processing of input query. And the domain-oriented knowledge base is used for identification of the domain.

Xu Yiqiu et al. proposed an interface design based on Ontology. It uses WordNet as the elementary lexicon and defines domain lexicon in addition to that. The language processing module and database processing module are linked by intermediate representation called Discourse Representation Structure(DRS). NLQ sentences are translate into DRS form, and then convert into SQL query. The Database Intelligent Querying System (DBIQS) is also an intermediate representation system which translates the NLQ to the intermediate form called Meaningful Representation. And then a query generator generates final SQL queries by mapping MR to semantic information [7].

However, users may have only a limited knowledge of the XML structure and may be unable to produce a correct XQuery expression, especially in the context of a heterogeneous information collection. The default is to use keyword-based search and we are all too familiar with how difficult it is to obtain precise answers by these means. We seek to address these problems by introducing the notion of Meaningful Query Focus (MQF) for finding related nodes within an XML document. MQF enables users to take full advantage of the preciseness and efficiency of XQuery without requiring (perfect) knowledge of the document structure [8][9]. Some of the approaches are given in this section. There are two main ideas in using keyword search for databases. First, sets of keywords expressed together in a query must match objects that are "close together" in the database (using some appropriate notions of "close together"). Second, there is a recognition that pure keyword queries are rather blunt – too many things of interest are hard to specify. The central idea in Schema-Free XQuery is that of a meaningful query focus (MQF) of a set of nodes. Beginning with a given collection of keywords, each of which identifies a candidate XML element to relate to, the MQF of these elements, if one exists, automatically finds relationships between these elements, if any, including additional related elements as appropriate. For example, for the query "Find the director of Gone with the Wind," there may be title of movie, and title of book with value "Gone with the Wind" in the database. However, we do not need advanced semantic reasoning capability to know that only movies can have a director and hence "Gone with the Wind" should be the title of a movie instead of a book. Rather, the computation of mqf(director, title) will automatically choose only title of movie, as this title has a structurally meaningful relationship with director. Furthermore, it does not matter whether the schema has director under movie or vice versa (for example, movies could have been classified based on their directors). In either case, the correct structural relationships will be found, with the correct director elements be returned. Schema-FreeXQuery greatly eases our burden in translating natural language queries in that it is no longer necessary to map the query to the precise underlying schema. We will use it as the target language of our translation process. The relationships between words in the natural language query must decide how the corresponding components in XQuery will be related to each other and thus the semantic meaning of the resulting query. We obtain such relationship information between parsed tokens from a dependency parser, which is based on the relationship between words rather than group of words [14] [15].

3. ANALYSIS OF PROBLEM

Though, existing system is using to query information from the database using natural language, there are some restrictions to this system. The limitations of existing system are as follows:

- The current system does not support the aggregate functions, grouping functions and arithmetic functions.
- There is scope of improvement in current system in terms of accuracy.

There are some other problems to achieve the goal of transforming natural language to SQL/XQuery such as automatically understanding natural language semantically but even syntactically is an open research problem. Second, even if we could fully understand any arbitrary natural language query, since this conversion requires mapping the understanding of intent into a specific database schema, translating this parsed natural language query into a correct formal query would remain an issue.

We will work extensively to overcome the limitations of existing system to achieve an anticipated goal.

REFERENCES

[1] "FLWOR - An Introduction to the XQuery FLWOR Expression." http:// www.stylusstudio.com/xqueryflwor.html. [September 2015].

[2] Yohan Chandra. "Natural Language Interfaces to Databases." http://www.ijritcc.org. [August 2015].

[3] "XML Tutorial." http://wws.w3schools.com. [August 2015].

[4] George Papamarkos, Lucas Zamboulis, Alexandra Poulovassilis."XMLDatabase."http://students.mimuw.edu.pl /pd291528/zbd/materialy/XmlDatabases.pdf. [August 2015].

[5] Y. Li, H. Yang, and H. Jagadish, "Nalix: A generic natural language search environment for xml data," ACM Transactions on Database Systems (TODS), vol. 32, no. 4, p. 30, 2007.

[6] Natural language Interface for Database: A Brief review Mrs. Neelu Nihalani, Dr. Sanjay Silakari , Dr. Mahesh Motwani. IJCSI International Journal of Computer Science March 2011 ISSN (Online): 1694-0814

[7] X. Wu, T. Ichikawa, and N. Cercone, "Natural language, knowledge, and database," in Knowledge-Base Assisted Database Retrieval Systems, pp. 1–13, World Scientific, 1996.

[8] Y. Li, I. Chaudhuri, H. Yang, S. Singh, and H. Jagadish, "Danalix: a domain-adaptive natural language interface for querying xml," in Proceedings of the 2007 ACM SIGMOD international conference on Management of data, pp. 1165–1168, ACM, 2007.

[9] H. J. Yunyao Li, "Schema-free xquery," in VLDB 04 Proceed- ings of the Thirtieth international conference on Very large data bases, pp. 72–83, IEEE, 2004.

[10] A. Shah, J. Pareek, H. Patel, and N. Panchal, "Nlkbidb-natural language and keyword based interface to database," in Advances in Computing, Communications and Informatics (ICACCI), 2013 International Conference on, pp. 1569–1576, IEEE, 2013.

[11] F. Li and H. Jagadish, "Constructing an interactive natural language interface for relational databases," Proceedings of the VLDB Endowment, vol. 8, no. 1, pp. 73–84, 2014.

[13] A. Giordani and A. Moschitti, "Semantic mapping between natural language questions and sql queries via syntactic pairing," in International Conference on Application of Natural Language to Information Systems, pp. 207–221, Springer, 2009.

[14] X. Yiqiu, W. Liwi, and Y. Shi, "The study on natural language interface of relational databases," in 2010 The 2nd Conference on Environmental Science and Information Application Technology, 2010.

[15] H. Li and Y. Shi, "A wordnet-based natural language interface to relational databases," in Computer and Automation Engineering (ICCAE), 2010 The 2nd International Conference on, vol. 1, pp. 514–518, IEEE,2010.