ISSN No:-2456-2165

An Efficient Way to Querying XML Database Using Natural Language

P. D. Thakare,
Head of the Department, Dept. of Computer Science,
Jagdamba College of Engineering Technology
Yavatmal, Maharashtra, India

Snehal Dhole,
Student, Dept. of Computer Science,
Jagdamba College of Engineering Technology
Yavatmal, Maharashtra, India

Abstract:- Information about anything plays a vital role in real world. In current era, databases and its related applications are best source to store information. Database technology is broadly used in many fields like healthcare, HRD, Intelligence services, Industries, service sector, Research and Development, Retail, Financial services, e-commerce, etc. When we store data to into database we need to retrieve the data and/or information from the database to use it. The SQL queries are followed by all the languages to retrieve data from database. But the challenge is everyone in real world does not know SQL queries. User find it very difficult to get data from the xml database using XML query languages. Retrieving data from database using human readable language or Native language like English is a convenient method instead of using query languages such as XQuery, SQL etc. Over the period of time, many intelligent natural language attachment to databases have been established, which caters tractable options for manipulating queries. Natural language Interface to Database is nothing but design theory of using Natural Language instead of SQL by processing. Here we proposed a system which will take English language sentences as input and will convert it into an XQuery/SQL expression. This XQuery/SQL statement can be assessed against an XML database. This query transcription is done by depicting the tokens in the dependency parse tree of the natural language query into the XQuery fragments. This paper introduces a diverse design which can interpret a varied choice of natural language including wh-question and also it will work for aggregate operations and s queries into formal database queries by attaining determined precision.

***Keywords: -** XML Database, XQuery, NLQ, XML, NLIDB.*

I. INTRODUCTION

In current era, Databases are capturing essential importance in a vast variety of application areas engaging private and public information systems. Databases are assembled with the objective of assisting the activities of data storage, processing, and reclaim associated with data management in information systems. Generally we used natural language, such as English to obtain the information. Not surprisingly, there have been several attempts made toward making of Natural language to fetch data from database. Nonetheless, two major hurdle underneath in the way of obtaining the eventual goal of support for arbitrary natural language queries: first, To understand natural language automatically is itself sound an open research problem, not just emphatically but even syntactically; second, even if we figure out any arbitrary natural language query, transferring this deciphered natural language query into a appropriate formal query would still be a problem since this translation needs mapping the understanding of language intent into a specific database schema. Due to the advancement and in depth applications of computer technologies, the popular applications of web technology in several areas, databases have become the storehouse of huge volumes of data. In relational databases, to recapture information from a database, one needs to generate a query in such way that the computer will comprehend and generate the desired output. The Structured Query Language (SQL) standard has been practiced in almost all languages for relational database systems. However, regaining data from these databases is hard job. To get Data from the database required understanding of SQL queries and databases. By looking at the use of widely spread of computer application it is very important to focus on the process of data captivating from database using natural language like English. Therefore the method of practicing natural language instead of SQL has given rise to the development of new type of processing method called Natural Language Interface to Database systems (NLIDB). NLIDB is a step towards the advancement of intelligent database systems (IDBS) to improve the users in achieving flexible querying in databases. NLIDB databases interface allow users to make use of database by using legitimate language like English to achieve conclusive result. NLIDB systems are also having some gains and weaknesses like other systems developed. Weakness is with its interface because we know the linguistic coverage is indefinite and we cannot forecast what type of query or question NLIDB can work with. Also to process linguistic operations into machine readable format will contribute in increase in time execution. NLIDB system is work in two stages which is also called as processing stages these are Linguistic processing and Database processing. In the Linguistic processing phase, natural language query (NLQ) mechanism is designed and construed into the corresponding database query fragments. Database ingress, internal operation like parsing and implementation work is carried out during the database processing stage, and then query is run by the system. To minimize this issue, We outline the concepts of token attachment and token relationship in natural language parse trees. The natural language sentences are transformed to XQuery. XQuery is a language utilized to discover and draw out elements and attributes from XML documents [1]. XQuery is used to get data from XML documents in same way the SQL does it for relational databases [2], [3]. An XPath expression is main component to build XQuery and all major databases support it [4]. To perform semantic grouping, we put forward the idea of core token as an operational mechanism and elected both query nesting and structural relationships between the outcomes. In addition to this one more point is that current system does not support the aggregate functions where user want to fetch the data from the database by providing input in term of aggregate function like Maximum and minimum etc.

II. EXSISTING WORK

In last decade, lot of ideas proposed in the field of natural language support to database. Nevertheless almost all designed idea that has been executed uses process of applying semantic and syntactic analysis to get a logical representation of the sentence accompanied by a conversion of the representation into a database query. There are few approaches which we listed down here. One of the approach is NALIX (Natural Language Interface for an XML Database) is an NLIDB system conceptualized and derived at the University of Michigan, Ann Arbor by Yunyao Li, Huahai Yang, and H. V.Jagadish (2006). The database utilized for this system is Extensible markup language (XML) database with Schema- Free XQuery which is used as the database query language. Schema-Free XQuery is nothing but a query language designed mainly for recovering information in XML. The plan is to search keyword for databases. However, flawless and clear keyword search certainly cannot be put forth. With collection of keywords, each keyword has number of candidate XML elements to put on record. All of these candidates are appended to MQF (Meaningful Query Focus), which will self-acting to find all the relations between these elements. The benefit of using Schema-Free Xquery is that we need not to insert or map query into database schema, because based on define keyword it will automatically discover relation between them. The complete processes of conversion take place in three phases: generating a parse tree, validating the parse tree, and translating the parse tree to an XQuery expression. NALIX can be classified as a syntax based system. Though, NALIX is varied from the syntax based design approaches because of the way it’s getting built. The NALIX execute a reversed engineering technique by structuring the system in such way that it will get data from a query language toward the sentences. Second approach is PRECISE. PRECISE is a system design evolved at the University of Washington by Ana-Maria Popescu, Alex Armanasu, Oren Etzioni, David Ko, and Alexander Yates (2004). The end database is in the type of a relational database using SQL as the query language. It put forth the idea of semantically yielding sentences which are sentences that can be converted to a unique semantic interpretation by examining some lexicons and semantic constraints. PRECISE was developed on two database domains. ATIS domain, which makes use of spoken questions about written forms, air travel and their correct paraphrasing in SQL query language. For example the GEOQUERY domain accommodate information about U.S. Geography. In GEOQUERY 77.5% question are semantically tractable. Using these questions provides PRECISE 100% authenticate and precious. The importance of PRECISE is based on the capability to match keywords in a sentence to the equivalent database structures. This procedure takes two phases to carry out this, first by confining the possible outcome using Maxflow algorithm and second by evaluating the syntactic structure of a sentence. That’s the reason why PRECISE is able to achieve powerfully in semantically tractable questions. On the other hand, NLIDB systems also has some disadvantages. The system compensates make most of recall to achieve gain accuracy in spite of it is able to gain high accuracy in semantically tractable questions. Another issue, it possess difficulty in handling nested structure as PRECISE adopts a heuristic based approach. In the system proposed by F.Siasar djahantighi et al. they established a technique which put together an genius system that can distinguish equivalent words in any language. It first try to find out the natural language sentences, and then converte that sentence to SQL queries. By using different kind of rules and semantic sets for all input and for all possible attribute a pre-processor is used. This help to translate a query to different form without affecting its meaning. It try to map the NLQ to SQL query by pairing the input with some particular patterns. Alessandra Giordani and Alessandro Moschitti cover this problem of transforming this natural sentence into query language semantics by the self-activating learning of a model from the lexical and syntactic representation of the training data samples. These are nothing but SQL queries and pairs of NL questions, which are illustrated using algorithm. in this system, support vector machines and Kernel approach are extensively utilized to showcase syntactic or semantic relationships enunciate by the pairs. Xu Yiqiu et al. put forth the idea of an interface principal design based on Ontology. It utilizes WordNet as the elementary lexicon. Also it defines domain lexicon in addition to that. Discourse Representation Structure (DRS) is the model that makes use to connection module i.e. language processing and database processing. NLQ sentences are deciphering into DRS form, and then convert into SQL query. The Database Intelligent Querying System (DBIQS) is system which convert the NLQ to the next intermediate type or form called Meaningful Representation. Here final SQL which is generated by querying for mapping MR to semantic information. D-HIRD is an NLIDB system developed by Rajender Kumal, Mohit Dul, and Shivani Jinda that obtain a NLQ in Hindi and then it produces corresponding SQL query, and provides the result by manipulating the data from respective databases. The system makes use of the Hindi Shallow Parser and processing of input query done by lexicon for linguistic.

III. PROPOSE WORK

In existing xquery generation will perform in the various way in which the Wh-word recognition, verb finding and other techniques are perform. In this some time in NL we can't predict the user input which will vary totally from user to user so it will show very variant results so that it is necessary to provide a way in which NL can be process with prediction based on apriori algorithm which will solve the problem of unrecognized queries found in implementation. So that the proposed work will help to improve the process of optimization in XQuery generation. The proposed system will make the XQuery Prediction if any word in NL will not get plotted to existing queries.

A. Tokanisation

Tokanisation is nothing but initial step where token conversion happens to discover the name tokens in the input sentence. In this phase the input string splits into token based on a regular expression. The regular expression syntax used is defined by XML Schema with a few modifications/additions in XQuery Path. Here token consist of phrase or word which is present in the sentence. These sentence try to compare and match to an element or an attribute in XML data set. It makes use of universal lexicon WordNet is used to locate such name tokens. The token conversation step is carried out to outline the tokens in the natural language query to respective XQuery components. The final outcome will be to try to leverage from NLQ tokens and connection marker to their respective XQuery components.

B. Apriori Algorithm

Apriori is used to manipulate on the databases which hold transaction of an algorithm. Apriori is an algorithm for usual association rule learning and item set mining over proceeding databases. It designed in such way that it discovered the frequently used respective items in the database and carry on them to maximum item sets as long as those item sets be evident to sufficiently contain in the database. It also uses other method which is designed in such way that data having no transaction for discovering association (Winepi and Minepi), or having no timestamps (DNA sequencing). These transaction apparently is an *item set*. Apart from this, Given a threshold, the Apriori algorithm discover the item sets which are part of larger group of at least transactions in the database. Apriori utilizes a "bottom up" design principle, where frequent subsets are spread out one item at a time where this step is known as *candidate generation.* It also consist of candidate groups which are tested against the data. The algorithm discontinued when no more successful extensions are found. Apriori utilizes search engine named as breadth-first search and a Hash tree structure totally candidate item sets efficiently. It produces candidate item sets of length from item sets of length. Then it trim the candidates which have an infrequent sub pattern. The candidate set consist of all frequent length item sets according to the downward closure lemma. It goes through the transaction database by scanning it to determine frequent item sets among the candidates. The entire algorithm can be divided into two steps:

Step 1: It comprise of minimum help to determine all the frequent sets with k items in a database. Step 2: It comprises by make use of the the self-join rule to determine the frequent sets with k+1 items with the help of frequent k-item sets. Reiterate this procedure from k=1 to the point when we are not able to inquire the self-join rule. Using this procedure of enlarging a frequent item set one at a time is called the bottom up approach.

At each step, the algorithm is expected to create the candidate sets from the large item sets of the preceding level, heading the downward closure lemma to get accesses of field of the data structure that showcase candidate set, which is partially take granted as zero. Many details are skip below, usually the most prime importance step of the implementation is the data structure. Data structure is used for storing the candidate sets, and counting their frequencies [6].

C. FlowChart

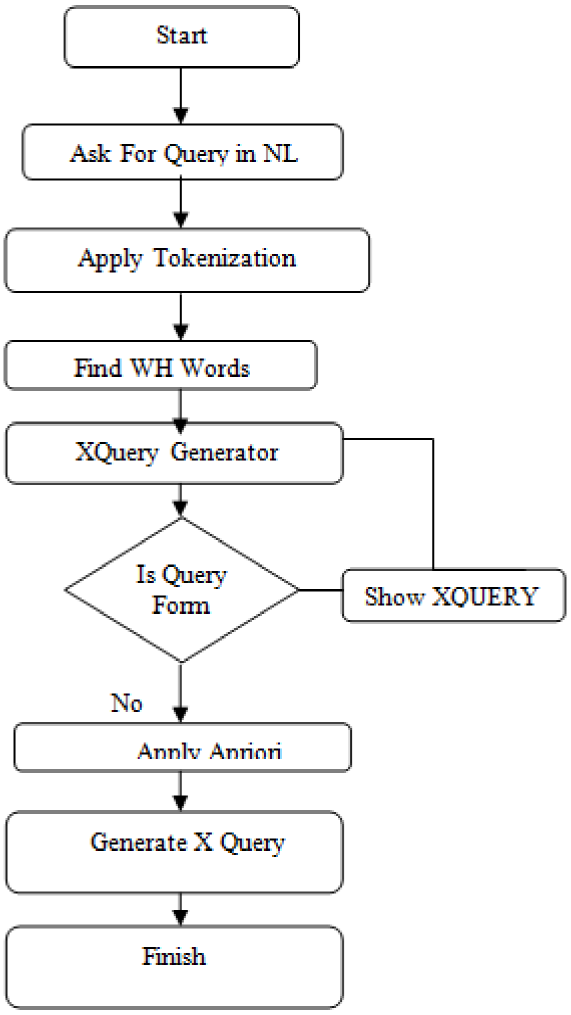

Fig 1

```
Apriori$(T, \epsilon)$
    $L_1 \leftarrow \{\text{large } 1 - \text{itemsets}\}$
    $k \leftarrow 2$
    while $L_{k-1} \neq \emptyset$
        $C_k \leftarrow \{c = a \cup \{b\} \mid a \in L_{k-1} \wedge b \notin a, \{s \subseteq c \mid |s| = k-1\} \subseteq L_{k-1}\}$
        for transactions $t \in T$
            $D_t \leftarrow \{c \in C_k \mid c \subseteq t\}$
            for candidates $c \in D_t$
                $count[c] \leftarrow count[c] + 1$
        $L_k \leftarrow \{c \in C_k \mid count[c] \geq \epsilon\}$
        $k \leftarrow k + 1$
    return $\bigcup_k L_k$
```

Fig 2:- Apriori Algorithm

D. Nesting Identification of Aggregate Functions

It takes into account input query sentence. If the input query sentence consists of function tokens like sum, average, number of etc. or quantifier tokens like every, some, all etc. mapping token. The mapping process is able to consider the nesting of XQuery fragments [10]. This Steps is useful to establish the nesting scope of aggregate functions, with respect to a particular variable that is attached with the function token. The aggregate functions is used in XQuery are: count, max, min and sum[11].

IV. CONCLUSION AND FUTURE WORK

As per the results given above it is seen that the proposed methodology for xml query generation plays an important role in in formation and generation of xquery by parsing the XML input given. The apriori algorithm make the execution in optimised way so that it will helpful for us in generation of xquery more effectively so that it is conclude that the proposed methodology is very helpfull for xquery generation.

In the proposed the apriori algorithm which will help for query generation will plays an important role it is also possible to implement the same work with advance apriori or some classification algorithm like naive baise classifier which may give the optimize way to generate the query with xml.

Query Type	***Query generated***	***Correct***
Wh-ques	99 Percent	95percent
Yes/No ques	99 Percent	95percent

Table 1:- Result Table

REFERENCES

[1]. Dr. Michael Kay, "FLWOR - An Introduction to the XQuery FLWOR Expression." http://www.stylusstudio.com/xquery-flwor.html.[September 2015].

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

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

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

[5]. "The Stanford Parser: A statistical parser." http://nlp.stanford.edu/software/lexparser.shtml. [August 2015].

[6]. "Apriori Algorithm", https://en.wikipedia.org/wiki/Apriori_algorithm

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

[8]. X. Wu, T. Ichikawa, and N. Cercone, "Natural language, knowledge, and database," in Knowledge-Base Assisted Database Retrieval Systems,

[9]. 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.

[10]. 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.

[11]. An Efficient Natural Language Interface to XML Database Jiffy Joseph Janu R Panicker janurpanicker@gmail.com