International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE)
ISSN: 0976-1353 Volume 24 Issue 4 – MARCH 2017.

Answering Best Possible Queries over XML Data

Shaikh Zubair Ahmed [#1] and Bazeem Ismaeil Khan [*2]

[#] *PG Student (ME-CSE), Everest Educational Society's College of Engineering and Technology, Aurangabad, India*

[*] *PG Student (ME-CSE), Everest Educational Society's College of Engineering and Technology, Aurangabad, India*

***Abstract*— Data exchange is the problem of finding an instance of a target schema, given an instance of a source schema and a specification of the relationship between the source and the target. Theoretical foundations of data exchange have recently been investigated for relational data. In this paper, we start looking into the basic properties of XML data exchange that is, restructuring of XML documents that conform to a source DTD under a target DTD, and answering queries written over the target schema. We define XML data exchange settings in which source-to target dependencies refer to the hierarchical structure of the data. Combining DTDs and dependencies makes some XML data exchange settings inconsistent. We investigate the consistency problem and determine its exact complexity. We then move to query answering, and prove a dichotomy theorem that classifies data exchange settings into those over which query answering is tractable, and those over which it is coNP-complete, depending on classes of regular expressions used in DTDs. Furthermore, for all tractable cases we give polynomial-time algorithms that compute target XML documents over which queries can be answered.**

Index Terms*— *XML, approximate query-answering, data mining, intentional information, succinct answers

I. INTRODUCTION

The In recent years the database research field has concentrated on XML (eXtensible Markup Language as a flexible hierarchical model suitable to represent huge amounts of data with no absolute and fixed schema, and a possibly irregular and incomplete structure. There are two main approaches to XML document access: keyword-based search and query-answering. The first one comes from the tradition of information retrieval, where most searches are performed on the textual content of the document; this means that no advantage is derived from the semantics conveyed by the document structure [1].

As for query-answering, since query languages for semi structured data rely the on document structure to convey its semantics, in order for query formulation to be effective users need to know this structure in advance, which is often not the case. In fact, it is not mandatory for an XML document to have a defined schema: 50% of the documents on the web do not possess one [2]. When users specify queries without knowing the document structure, they may fail to retrieve information which was there, but under a different structure. This limitation is a crucial problem which did not emerge in the context of relational database management systems [3].

Frequent, dramatic outcomes of this situation are either the information overload problem, where too much data are included in the answer because the set of keywords specified for the search captures too many meanings, or the information deprivation problem, where either the use of inappropriate keywords, or the wrong formulation of the query, prevent the user from receiving the correct answer. As a consequence,when accessing for the first time a large dataset, gaining some general information about its main structural and semantic characteristics helps investigation on more specific details [4].

This paper addresses the need of getting the gist of the document before querying it, both in terms of content and structure. Discovering recurrent patterns inside XML documents provides high-quality knowledge about the document content: frequent patterns are in fact intentional information about the data contained in the document itself, that is, they specify the document in terms of a set of properties rather than by means of data. As opposed to the detailed and precise information conveyed by the data, this information is partial and often approximate, but synthetic, and concerns both the document structure and its content. In particular, the idea of mining association rules to provide summarized representations of XML documents has been investigated in many proposals either by using languages and techniques developed in the XML context, or by implementing graph- or tree-based algorithms [5].

In this paper we introduce a proposal for mining and storing TARs (Tree-based Association Rules) as a means to represent intentional knowledge in native XML. Intuitively, a TAR represents intentional knowledge in the form SB ⇒ SH, where SB is the body tree and SH the head tree of the rule and SB is a sub tree of SH. The rule SB ⇒ SH states that, if the tree SB appears in an XML document D, it is likely that the "wider" (or "more detailed"), tree SH also appears in D. The intentional information embodied in TARs provides a valid support in several cases: 1) It allows to obtain and store

implicit knowledge of the documents, useful in many respects: (i) when a user faces a dataset for the first time, s/he does not know its features and frequent patterns provide a way to understand quickly what is contained in the dataset; (ii) besides intrinsically unstructured documents, there is a significant portion of XML documents which have some structure, but only implicitly, that is, their structure has not been declared via a DTD or an XML-Schema [6].

Since most work on XML query languages has focused on documents having a known structure, querying the above-mentioned documents is quite difficult because users have to guess the structure to specify the query conditions correctly. TARs represent a data guide that helps users to be more effective in query formulation; (iii) it supports query optimization design, first of all because recurrent structures can be used for physical query optimization, to support the construction of indexes and the design of efficient access methods for frequent queries, and also because frequent patterns allow to discover hidden integrity constraints, that can be used for semantic optimization; (iv) for privacy reasons, a document answer might expose a controlled set of TARs instead of the original document, as a summarized view that masks sensitive details [7].

TARs can be queried to obtain fast, although approximate, answers. This is particularly useful not only when quick answers are needed but also when the original documents are unavailable. In fact, once extracted, TARs can be stored in a (smaller) document and be accessed independently of the dataset they were extracted from. Summarizing, TARs are extracted for two main purposes: 1) to get a concise idea – the gist – of both the structure and the content of an XML document, and 2) to use them for intentional query answering, that is, allowing the user to query the extracted TARs rather than the original document. In this paper we concentrate mainly on the second task [8].

We have applied our techniques in the Odyssey EU Project1, whose objective is to develop a platform for automated sharing, management, processing, analysis and use of ballistic and crime scene information across Europe. Frequent patterns, in the form of TARs, provide summaries of these integrated datasets shared by different EU Police Organizations. By querying such summaries, investigators obtain initial knowledge about specific entities in the vast dataset(s), and are able to devise more specific queries for deeper investigation. An important side-effect of using such a technique is that only the most promising specific queries are issued towards the integrated data, dramatically reducing time and cost [9].

This paper provides a method for deriving intentional knowledge from XML documents in the form of TARs, and then storing these TARs as an alternative, synthetic dataset to be queried for providing quick and summarized answers. Our procedure is characterized by the following key aspects: a) it works directly on the XML documents, without transforming the data into any intermediate format, b) it looks for general association rules, without the need to impose what should be contained in the antecedent and consequent of the rule, c) it stores association rules in XML format, and d) it translates the queries on the original dataset into queries on the TARs set. The aim of our proposal is to provide a way to use intentional knowledge as a substitute of the original document during querying and not to improve the execution time of the queries over the original XML dataset, like in [10].

The remainder of this paper is organized as in the following sections. Section 2will describe the related workson data exchange over xml queries. Section 3 will present the proposeddata exchange over xml queries method. In Section 4, we will analyzethe results ofproposed method and compare it with standard data exchange methods. Finally, a brief conclusion will be given in Section 5.

II. RELATED WORK

Hovy et al. describes a set of heuristics that researchers at ISI/USC used for semi-automatic alignment of domain ontology to a large central ontology. Their techniques are based mainly on linguistic analysis of concept names and natural-language definitions of concepts. (There is a limited use of taxonomic relationships as well). First, the matcher uses natural-language–processingtechniques to split compositebyword names (a common occurrence in concept names). It then compares substrings of different lengths to find concept names that are similar to each other. The second consideration is the words used in natural-language definitions of concepts. The matcher compares the number and the ratio of shared words in the definitions to find definitions that are similar. An experimentally determined formula for combining these measures of similarity yields potential matchers that the user needs to examine and approve [11].

Navatheet al presented that, as databases become widely used, there is a growing need to translate data between multiple databases. This problem arises when organizations consolidate their databases and hence must transfer data from old databases to the new ones. It forms a critical step in data warehousing and data mining, two important research and commercial. In these applications, data coming from multiple sources must be transformed to data conforming to a single target schema to enable further data analysis [12].

In recent years, the explosive growth of information online has given rise to even more application classes that require semantic integration. One application class builds data-integration systems. Such a system provides users with a uniform query interface (called mediated schema) to a multitude of data. In general, path indexes are proposed to quickly answer queries that follow some frequent path template, and are built by indexing only those paths having highly frequent queries. We start from a different perspective: we want to provide a quick, and often approximate, answer also to casual queries [13].

Inokuchi et al. presented a critical problem in building a data-integration system, therefore, is to supply the semantic matches. Since in practice data sources often contain duplicate items another important problem is to detect and eliminate duplicate data tuples from the answers returned by the sources before presenting the final answers to the user query. Another important application class is peer data management, which is a natural extension of data integration [14].

Goldman et al. presented a peer data management system

does away with the notion of mediated schema and allows peers (that is, participating data sources) to query and retrieve data directly from each other. Such querying and data retrieval require the creation of semantic correspondences among the peers. Recently there has also been considerable attention on model management, which creates tools for easily manipulating models of data (for example, data representations, website structures, and entity relationship [ER] diagrams). Here semantic integration plays a central role, as matching and merging models form core operations in model management algebras [15].

Washio et al. presented a data-sharing application arise in numerous current real-world domains. They also play an important role in emerging domains such as e-commerce, bioinformatics, and ubiquitous computing. Some recent developments should dramatically increase the need for and the deployment of applications that require semantic integration. The Internet has brought together millions of data sources and makes possible data sharing among them. The widespread adoption of XML as a standard syntax to share data has further streamlined and eased the data-sharing process. The growth of the semantic web will further fuel data-sharing applications and underscore the key role that semantic integration plays in their deployment [16].

III. PROPOSED WORK

The proposed work aims to provide generic support for querying provenance information to enable a wide range of users and applications. Common types of provenance queries we want to support include standard lineage queries for determining the data and invocations used to derive other data; queries that allow users to ensure that specific data and invocation dependencies were satisfied within a run; queries for determining the inputs and outputs of invocations (e.g., based on the actors used and their parameters); and queries. The proposed system consists of the following four modules.

1. The System Construction Module
2. Query Relaxations
3. Approximate Queries Processing
4. top-*k* Retrieval Approach

A probabilistic XML document defines a probability distribution over a space of deterministic XML documents. Each deterministic document belonging to this space is called a possible word. A document represented as a labeled tree has ordinary and distributional nodes. Ordinary nodes are regular XML nodes and they may appear in deterministic documents, while distributional nodes are only used for defining the probabilistic process of generating deterministic documents and they do not occur in those documents. As we adopt PrXML{ind,mux} as the probabilistic XML model, two types of distributional nodes, IND and MUX, may appear in a p-document. The architecture of the proposed work is given in the figure 1.

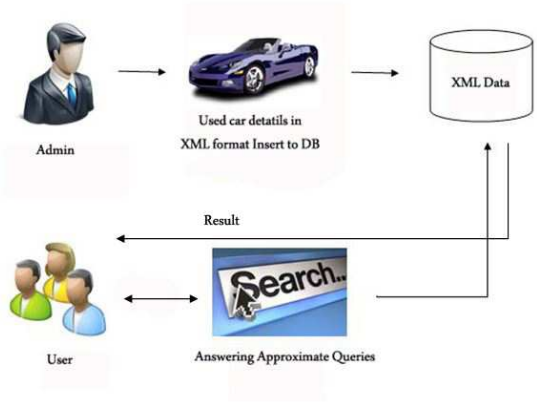

Figure 1. Proposed System Architecture

A. *System Construction Module*

In the first module, we develop our proposed system with the entities, to show the performance of our contribution model. We consider a data model for XML where information is represented as a series of data trees. Essentially, a data tree represents a portion of the real world through entities (usually contains a set of attributes), values, and relationships among them. A simple XML data instance which contains a heterogeneous collection of used cars.We develop the system for the application of user car sales system. The entities available in our system are admin and users. A organizes cars based on the model of a car, B organizes cars according to the selling location, and C includes cars that are organized by *model* and *year*. The approximate queries can be achieved by introducing substitutes having the approximate query intents with the original query, which we call similar substitutes.

B. *Query Relaxations:*

Framework of query relaxations for supporting approximates the queries over XML data. The answers underlying this framework are not compelled to strictly satisfy the given query formulation; instead, they can be founded on properties inferable
from the original query**.** A query relaxation method incorporating not only structures and contents, but also the factors that users are more concerned about (we infer these factors by first analyzing the original query and then identifying relaxation ordering of structures and nodes), to answer approximate XML queries.

Our method surmises the factors that users are more concerned about based on the analysis of user's original query for supporting query relaxations. In addition, our approach differentiates the relaxation ordering instead of giving an equal importance to each node to be relaxed. In particular, the first relaxed structure to be considered is the one that has the highest similarity coefficient with original query, and the first node to be relaxed is the least important node.

Query relaxation enables systems to weaken the query constraints to a less restricted form to accommodate users' needs. Traditionally, queries submitted by users are modified in various aspects and ways to cope with different situations. The importance of such techniques that enable automatic query modification stems from the fact that this behavior is a very common activity in human discourse.

C. Approximate Query Processing:

In this module Approximate query processing (AQP) is an alternative way that returns approximate answer using information which is similar to the one from which the query would be answered.We first propose a sophisticated framework of query relaxations for supporting approximate queries over XML data. The answers underlying this framework are not compelled to strictly satisfy the given queryformulation; instead, they can be founded on properties inferablefrom the original query.

The approximate queries can be achieved by introducing substitutes having the approximate query intents with the original query, which we call similar substitutes.Approximate query is a retrieval technique, which finds matches that are likely to be relevant to a search argument even when the argument does not exactly correspond to the desired information. An approximate query is done by means of an approximate matching strategy, which returns a list of results based on likely relevance even though search argument may not exactly match.

D. Top-k Retrieval Approach:

In this module a novel top-*k* retrieval approach that can smartly generate the most promising answers in an order correlated with the ranking measure.The proposed similarity assessment and the degrees of importance we complement the query relaxations with an automatic retrieval approach that can efficiently generate the most promising top-*k* answers.The answer score of an answer (a match) measures the relevance of that answer to the user's query. For a given parameter *k*,the top-*k* problem is searching the best top-*k* answers (matches) ordered from best (highest answer score) to the worst.

IV. EXPERIMENTAL ANALYSIS

We have developed a prototype system supporting MFA's and algorithms rewrite and HyPE (and its variants OptHyPE and OptHyPE-C). In our experiments, we focused on the most time-consuming module of SMOQE, i.e., the query evaluator. The experiments were conducted on a dual 2.3GHz Apple Xserve with 4GB of memory. For the generation of our datasets, we used ToXGene. We generated XML documents that conform to our recursive hospital DTD, with sizes ranging from 7MB to 70MB, in 7MB increments. Each increment roughly corresponds to adding the medical history of 10,000 patients to our document tree. Therefore, the largest document stores the medical history of approximately 100,000 patients. The maximal depth of the trees is 13.

The generated data consist mainly of element nodes, and to a lesser extent of text nodes. Therefore, the size of the document has a direct impact on query evaluation. For example, our smallest document (7MB) consists of 303,714 element nodes vs 151,187 text nodes. The text nodes are used to increase the selectivity of queries but their size is kept to a minimum (so as not to increase the document size). Using the generated document trees, we conducted two sets of experiments, one regarding XPath evaluation, the other regarding regular XPath. The reported times are averaged over at least 5 runs of each experiment.

Since regular XPath subsumes XPath, we investigate the performance of HyPE and its variants for the evaluation of XPath queries. We compared our performance with that of the Java API for XML Processing Reference Implementation, which relies on XERCES and XALAN. We also compared with JAXP-COMPILE, a version of JAXP that precompiled the input query and converts it into a set of Java classes. The two JAXP versions had similar performance and thus we only report one of them. We ran various types of XPath queries with simple filters on data values, unions of queries, and Boolean combinations of filters.

We show the evaluation time both for queries with result sizes of a few hundreds of nodes and queries that return a few thousands of nodes For each query type, we report the evaluation time for JAXP, HyPE, OptHyPE and OptHyPE-C. The figures show clearly that our algorithm consistently outperform JAXP by a factor of three for HyPE, and four for OptHyPE and OptHyPE-C. We also observe that in most cases, both optimized versions of HyPE run almost twice as fast as HyPE. Note as well that the performance of OptHyPE-C is almost identical to that of OptHyPE (while OptHyPE-C uses a compressed index). The figure 2, 3, 4 shows the performance of the proposed system in comparison with standard methods.

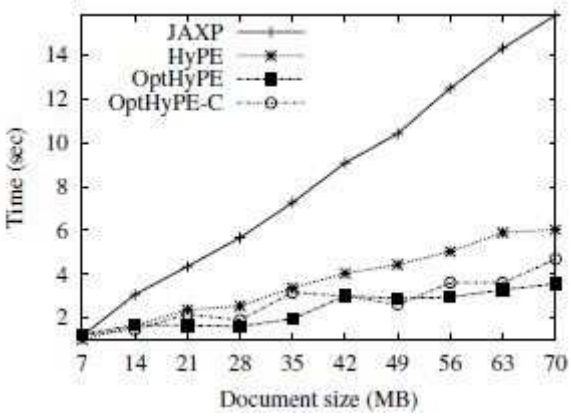

Figure 2. A filter returning a large set of nodes

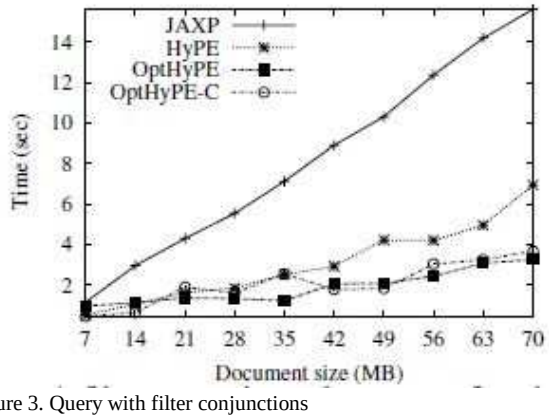

Figure 3. Query with filter conjunctions

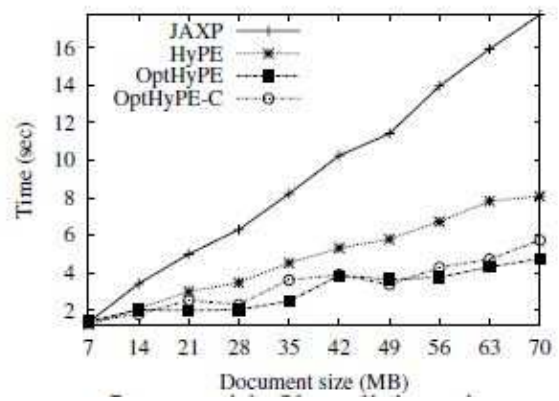

Figure 4.Query with filter disjunctions

The second set of experiments investigated the performance of evaluating regular XPath queries with the different versions of HyPE. Existing alternatives rely on a translation of regular XPath into a more powerful query language like XQuery. We conducted a series of experiments following this approach. Specifically, we translated several regular XPath queries into XQuery and evaluated them in GALAX.

These experiments consistently showed that the queries in XQuery required considerably more time than their regular XPath counterparts. As a result we omit GALAX from our discussion because even for a simple regular XPath query on the smallest used document tree, GALAX needed more time than HyPE for the same query on the largest tree. Hence, we only focus on the relative performance of our algorithm. We ran different types of regular XPath queries that involve Kleene star outside a filter, inside a filter, filters inside Kleene stars and combinations thereof

The overall conclusion is consistent with our observations regarding XPath queries. Indeed, OptHyPE and OptHyPE-C show considerable improvement over HyPE. An interesting observation is that HyPE prunes a substantial number of element nodes. Specifically, HyPE (resp. OptHyPE) prunes, on average, 78.2% (resp. 88%) of the element nodes for our example queries.

V. CONCLUSION

In this paper, we present a novel datagathering scheme for WSNs with a single mobile sink called, virtual binary-tree infrastructure based data gathering scheme, which constructs a virtual binary-tree infrastructure for sensor nodes to inquiry the sink location. The sink moves along the peripheral leaf-areas in a clockwise manner and stops in each leaf-area to collect data. The sojourn time in different areas is based on the total amount of data collected after the sink enters each area. The main contribution of our virtual binary-tree infrastructure based data gathering scheme is to collect the whole network data with less broadcast overhead efficiently. We also investigate the impact of different structures on the average energy consumption, the maintenance cost, the packet loss rate and the number of packets collected, respectively. And compare the proposed scheme with standard data collection schemes in the simulation. Results show that the agent based data gathering scheme is energy efficient and prolongs the network lifetime significantly with tolerable packet loss rate, especially in large-scale intensive networks.

REFERENCES

[1] Gary Marchionini. Exploratory search: from finding to understanding.Communications of the ACM, 49(4):41–46, 2006.
[2] T. Asai, K. Abe, S. Kawasoe, H. Arimura, H. Sakamoto, and S. Arikawa. Efficient substructure discovery from large semi-structured data. In Proc. of the SIAM Int. Conf. on Data Mining, 2002.
[3] T. Asai, H. Arimura, T. Uno, and S. Nakano. Discovering frequent substructures in large unordered trees. In Technical Report DOI-TR 216, Department of Informatics, Kyushu University. http://www.i.kyushuu.ac.jp/doitr/trcs216.pdf, 2003.
[4] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In Proc. of the 20th Int. Conf. on Very Large Data Bases, pages 487–499. Morgan Kaufmann Publishers Inc., 1994.
[5] World Wide Web Consortium. XQuery 1.0: An XML query language, 2007. http://www.w3C.org/TR/xquery.
[6] A. Termier, M. Rousset, M. Sebag, K. Ohara, T. Washio, and H. Motoda. Dryadeparent, an efficient and robust closed attribute tree mining algorithm. IEEE Transactions on Knowledge and Data Engineering, 20(3):300–320, 2008.
[7] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In Proc. of the 8th ACM Int. Conf. on Knowledge Discovery and Data Mining, pages 217–228, 2002.
[8] T.K Srinath and Joby George, Data Mining for Xml Query Answering Support,In International Journal of Scientific and Research Publications, Volume 5, Issue 11, November 2015
[9] Mirjana Mazuran, Elisa Quintarelli, and Letizia Tanca, Data mining for XML query-answering support, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, 2011
[10] K. Wong, J. X. Yu, and N. Tang. Answering xml queries using pathbased indexes: A survey. World Wide Web, 9(3):277–299, 2006
[11] E. Hovy. Combining and standardizing largescale, practical ontologies for machine translation and other uses. In The First International Conference on Language Resources and Evaluation (LREC), pages 535–542, Granada, Spain, 1998.
[12] Navethe T, KruzanskiAdome, Advances in frequent itemset mining implementations: report on FIMI'03. SIGKDD Explorations, 6(1):109– 117, 2004.
[13] A. Jimenez, F. Berzal, and J. C. Cubero. Mining induced and embedded ´ subtrees in ordered, unordered, and partially-ordered trees. In Proc. of the 17th Int. Symposium on Methodologies for Intelligent Systems, pages 111–120, 2008.
[14] J. Widom. Dataguides: Enabling query formulation and optimization in semistructured databases. In Proc. of the 23rd Int. Conf. on Very Large Data Bases, pages 436–445, 1997.
[15] R. Goldman and J. Widom. Approximate DataGuides. In Proc. of the Workshop on Query Processing for Semistructured Data and NonStandard Data Formats, pages 436–445, 1999.
[16] T. Washio, and H. Motoda. Complete mining of frequent patterns from graphs: Mining graph data. Machine Learning, 50(3):321– 354, 2003.
[17] D. Katsaros, A. Nanopoulos, and Y. Manolopoulos. Fast mining of frequent tree structures by hashing and indexing. Information & Software Technology, 47(2):129–140, 2005.
[18] M. Kuramochi and G. Karypis. An efficient algorithm for discovering frequent subgraphs. IEEE Transactions on Knowledge and Data Engineering, 16(9):1038–1051, 2004.
[19] H. C. Liu and J. Zeleznikow. Relational computation for mining association rules from xml data. In Proc. of the 14th ACM Conf. on Information and Knowledge Management, pages 253–254, 2005.
[20] Gary Marchionini. Exploratory search: from finding to understanding. Communications of the ACM, 49(4):41–46, 2006.