Volume-6, Issue-1, January - 2017 • ISSN No 2277 - 8160 IF : 3.62 | IC Value 80.26

Original Research Paper Computer Science

Mixed Operation For Query Processing Over XML

kanchankumari MCA Research scholler, Department of Statistic and Computer Applications, Tilkamajhi Bhagalpur University, Bhagalpur.

SudhanshuShekhar MCA,LLB, Department of Statistic and Computer Applications, Tilkamajhi Bhagalpur University, Bhagalpur.

B.k.Das Associate professor, Department of Statistic and Computer Applications, Tilkamajhi Bhagalpur University, Bhagalpur.

ABSTRACT *Query routing and Query processing are three main phase discused and vital role in query processing time. Locating relevant data sources and generating a close to optimal execution plan become more difficult.* ***In this paper,*** *we concentrate our study on proposed solutions for* ***Fundamental XML and elements, Query Processing Architecture,query processing time over system*** *problems. The focus in such systems is how to carry out an efficient query routing in order to find the nodes storing a desired file. Recently, several research works have been made to extend P2P systems to be able to share data having a fine granularity (i.e. atomic attribute) and to process queries written with a highly expressive language (i.e. SQL).Sharing musical files via the Internet was the essential motivation of early P2P systems. Despite of the great success of the P2P file sharing systems, these systems support only "simple" queries. These works have led to the emergence of P2P data sharing systems that represent a new generation of P2P systems and, on the other hand, a next stage in a long period of the database research area. The characteristics of P2P systems (e.g. large-scale, node autonomy and instability) make impractical to have a global catalog that represents often an essential component in traditional SQL, RDBMS, XML database systems. Usually, such a catalog stores information about data, schemas and data sources.*

KEYWORDS :

1. INTRODUCTION

The more the resources are available in a P2P system, the more the computing power and the storage capacity have important values. This advantage enables P2P systems to perform complex tasks with relatively low cost without any need to powerful servers. In the next section, we highlight the notion of ***"Mixed operation Systems".***

Peers are connected through a logical network topology implemented on top of an existing physical network, which may dynamically adapt to cope with peers joining and departing, as well as with network and peer failures the P2P model has emerged as a new and popular computing model for many Internet applications, such as le sharing, collaborative computing, and instant messaging. [6] A P2P network consists of a large number of nodes, called peers, that share data and resources with other peers on an equal basis .In contrast to traditional client-server architectures, a node in a P2P network can act as both a service provider and a client. [8] Compared to traditional client-server systems, P2P systems are more scalable, Exible, fault-tolerant, and easier to deploy.

Nowadays, Peer-to-Peer (hereafter P2P) systems become very popular. This popularity can be seen as a result of the features of these systems such as: scalability, node autonomy, self-configuration and decentralized control. P2P systems offer a good opportunity to overcome the limitations of the Client/Server based systems. [7] By avoiding bottlenecks and being fault tolerant, P2P systems are suitable for large-scale distributed environments in which nodes (interchangeably called peers) can share their resources (e.g. computing power, storage capacity, network bandwidth) in an autonomously and decentralized manner.

2. Fundamental XML and elements

After relational, network-based, hierarchical, object-oriented, object-relational, and deductive database systems, academic research and businesses increase their attention to the database-driven processing of XML documents, resulting in a new kind of information system, namely the *(native) XML database system* (XDBS).[10] This devel-opment is reasonable, because the *eXtensible Markup Language* nowadays plays an important role in various key technologies like content management systems, electronic data interchange, and data integration techniques. Furthermore, for the management of a possibly large collection of XML documents, the classical advantages of dedicated database systems over file systems still hold: Convenient use of XML data through a standardized application programming interface (API); Transactional warranties for all operations on XML data; Processing of large volumes of data, measured in number of documents as well as document size. Further advantages of database systems like scalability with respect to the current transactional load, high availability and fault tolerance, as well as data and application independence shall be mentioned for completeness, though they are not XML specific.

In [1] Michael Haustein and Theo Härder outline the design and realization for a subset of these desirable concepts—a prototype for academic research and native XDBS, named *XML Transaction Coordinator* (XTC). Currently, XTC provides an internal node interface called taDOM, which includes the features of the Document Object Model enhanced with user transactions. Every sequence of DOM operations can be encapsulated into a transaction and can thus benefit from the ACID warranties. For declarative language access, an XQuery processor resides on top of this interface. Its implementation follows the concepts given in the XQuery formal semantics [2], thereby neglecting important optimization techniques, which were crucial for the success of relational database systems in the past thirty years.

Throughout this paper, we focus on the problem of query evaluation for declarative XDBS access using XQuery. Our contribution can be understood as a road-map that reveals a desirable set of functionalities for the XTC query processor.

2.1. XML Documents

An *XML document* can be seen as a rooted, ordered, labelled tree, where each node corresponds to an element or a value, and the edges represent (direct) element-subelement or element-value

relationships. The ordering of sibling nodes (children of the same parent node) implicitly defines a total order on the nodes in a tree, obtained by traversing the tree nodes in preorder. An XML database can be viewed as an XML document, once a dummy root node has been added to convert the forest into a tree.

Example 1 *Figure 1 shows a fragment of an XML docu-ment that specifes information about a book. The gure shows four children of the* book *root node, namely:* title, all authors, year, *and* chapter, *in that order. Intuitively,we can interpret the XML fragment in the gure as a book published in the year* 2000 *by* Jane Poe, John Doe, *and* Jane Doe, *entitled* XML. *Its rst chapter, also entitled* XML,*starts with the section* Origins.

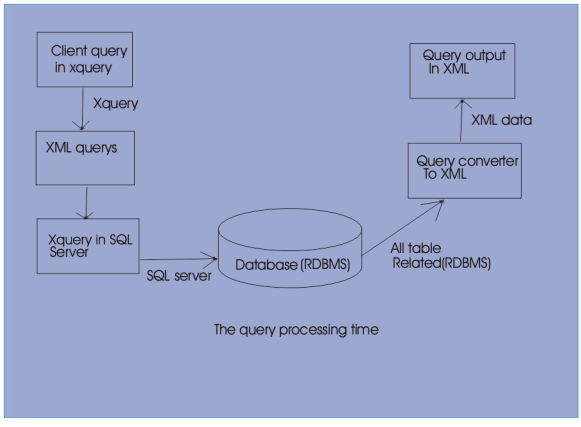
3. Query Processing Architecture

In this section, we present our high-level query-processing architecture. We begin by illustrating how XML views are created and queried in XPERANTO.

As a starting point, XPERANTO automatically creates a *default XML view,* which is a low-level XML view of the underlying relational database. Users can then define their own views on top of the default view using XQuery. Moreover, views can be defined on top of views to achieve higher levels of abstraction. The main advantage of this approach is that a standard XML query language is used to create and query views. This is in contrast to approaches such as [3][4][5], where a proprietary language is used to define the initial XML view of the underlying relational database.

Figure 1 shows the default XML view for a simple purchase-order database. As shown, the database consists of three tables, one table to keep track of customer orders, a second table to keep track of the items associated with an order, and a third table to keep track of the payments due for each order. Items and payments are related to orders by an order id(oid). In the default XML view, top-level elements correspond to tables with table names appearing as tags (there is no specific ordering among the

4. Query processing time over system



The primary query path which is taken for all queries on the data. if a workload query can be answered by the materialized XML cluster then only the primary path is taken otherwise the query next follows the secondry query path,the input query is pushed down to the relational level and is answered using the stored relations rather then the materialized XML.

The xml clusters are stored as values of an attributed in a special relation. The system queries the relation in SQL to find the most relevant cluster and then poses the XQuery query on the cluster. The schema for the clusters is specified by the database administrator.

- **Conclusions**

In this paper, we extend the relationships all steps query processing time where every steps important and cyclic and this cyclic satisfied throw ***Fundamental XML and elements, Query Processing Architecture, query processing time over system*** *and this throw, we are ease accessing output and understand what happen when we are query on server and get the output and before its throw we are know what are steps actully.*

References

1. Michael P. Haustein. Eine XML-Programmierschnittstelle zur transaktionsgeschützten Kombination von DOM, SAX und XQuery. In 11.GI-Fachtagung für Datenbanksysteme in Business, Technologie und Web (BTW) , 2005. http://wwwdvs.informatik.uni-kl.de/pubs/papers/Hau04.BTW.html.
2. Denise Draper, Peter Frankhauser, Mary Fernández, Ashok Malhotra, Kristoffer Rose, Michael Rys, Jérôme Siméon, and Philip Wadler. XQuery 1.0 and XPath 2.0 Formal Semantics. Technical report, World Wide Web Consortium (W3C), 2004. http://www.w3.org/TR/xquery-semantics/.
3. J. Cheng, J. Xu, "XML and DB2", ICDE Conf., San Diego, March 2000, pp. 569-573.
4. M. Fernandez. W. Tan, D. Suciu, "SilkRoute: Trading Between Relations and XML", World Wide Web Conf., Toronto, Canada, May 1999.
5. Microsoft Corp. http://www.microsoft.com/XML.
6. L. Galanis, Y. Wang, S. R. Je ery, and D. J. DeWitt. Locating Data Sources in Large Distributed Systems. VLDB 2003.
7. A.Y. Halevy, Z.G. Ives, J. Madhavan, P. Mork, D. Suciu, and I. Tatarinov. The Piazza Peer Data Management System. IEEE Trans. Knowl. Data Eng. 16(7): 787-798 (2004).
8. R. Huebsch, et al. The Architecture of PIER: an Internet-Scale Query Processor. CIDR 2005.
9. G. Koloniari and E. Pitoura. Content-Based Routing of Path Queries in Peer-to-Peer Systems. EDBT 2004.