Relationally Mapping XML Queries For Scalable XML Search

Rebecca J. Cathey, Steven M. Beitzel, Eric C. Jensen, David Grossman, Ophir Frieder
Information Retrieval Laboratory
Illinois Institute of Technology
{cathey, beitzel, jensen, grossman, frieder}@ ir.iit.edu

***Abstract*—The growing trend of using XML to share security data requires scalable technology to effectively manage the volume and variety of data. Although a wide variety of methods exist for storing and searching XML, the two most common techniques are conventional tree-based approaches and relational approaches. Tree-based approaches represent XML as a tree and use indexes and path join algorithms to process queries. In contrast, the relational approach seeks to utilize the power of a mature relational database to store and search XML. This method relationally maps XML queries to SQL and reconstructs the XML from the database results.**

We use the XBench benchmark to compare the scalability of the SQLGenerator, our relational approach, with eXist, a popular tree-based approach.

I. Introduction

XML is a flexible and powerful tool that enables vital security sharing in heterogeneous environments [1]. Since XML can be extended to include domain specific tags, information can be encoded with meaningful structure and semantics that allow rapid information sharing among devices and organizations. We examine the conventional tree approach and the relational mapping of XML queries to determine which method has the potential to search large collections of XML data.

We used XBench (http://db.uwaterloo.ca/ddbms/projects/xbench/index.html) to create a heterogeneous collection of multiple schema data-centric XML documents. We generated an 8GB collection from the modified XBench templates. The 500MB, 1GB, 2GB, and 4GB collections were created from random subsets of the 1GB, 2GB, 4GB, and 8GB collections respectively.

XBench provides a set of twenty queries that challenge a system with XML-specific features as well as conventional functionalities. The XML features covered by the queries include exact match (Q1), ordered access (Q5), quantifier expressions (Q6, Q7), regular path expressions (Q8, Q9), sorting document construction (Q10, Q11), retrieving individual documents (Q12), and text search (Q17).

II. Results

Two XML retrieval systems are used: the SQLGenerator (http://www.ir.iit.edu/projects/SQLGenerator.html) and eXist 1.0. The SQLGenerator uses a model mapping relational approach [2] while eXist uses an XML-specific B+-tree indexing approach [3]. The SQLGenerator was run using MySQL version 4.1.11.

In Fig. 1, we show the total query time for each query. Also shown are the mean and total times for each run. All timings given represent the average execution time of the queries (in random order) over five runs. To ensure a cold cache, the server was rebooted between runs. Overall, the relational approach outperformed the tree-based approach on all five collections. However, the tree based approach outperformed the relational approach for both quantifier queries. On average, the relational approach took 17.5 times as long to execute the 8GB queries than the 500MB queries. This is very close to the expected linear scaling factor of 16. On the other hand, the tree based approach took 55.6 times as long to execute the queries on the 8GB collection.

	Collection Size									
	500MB		*1GB*		*2GB*		*4GB*		*8GB*	
	tree	rel	tree	rel	tree	rel	tree	rel	tree	rel
Q1	15.52	0.43	26.75	0.44	59.41	0.51	131.11	0.71	484.24	0.67
Q5	14.52	0.77	25.72	0.78	56.20	0.96	129.61	1.27	498.83	1.47
Q6	2.03	31.08	4.64	50.26	18.97	52.37	28.73	183.10	TIMEOUT	466.58
Q7	2.85	25.0	6.43	38.88	10.32	71.70	26.94	243.15	217.72	594.19
Q8	14.57	0.58	25.89	0.51	56.52	0.69	131.00	0.87	478.43	1.10
Q9	14.73	0.40	25.38	0.41	53.50	0.46	124.22	0.63	426.58	0.72
Q10	16.22	1.16	29.89	1.51	68.35	2.73	139.53	5.97	1,850.99	13.22
Q11	16.75	0.68	29.73	0.76	65.58	1.46	137.04	2.81	1,829.84	6.17
Q12	14.35	0.64	25.41	0.63	54.03	0.79	129.80	0.89	458.05	1.06
Q16	14.41	0.58	25.11	0.52	51.29	0.70	125.06	0.76	440.00	0.80
Q17	32.33	1.78	53.14	2.25	97.53	19.50	638.29	37.29	1,317.18	21.86
mean	**14.39**	**5.74**	**25.27**	**8.81**	**53.79**	**13.81**	**158.29**	**43.40**	**800.18**	**100.71**
total	**158.28**	**63.09**	**278.09**	**96.94**	**591.7**	**151.88**	**1741.33**	**477.44**	**8001.86**	**1107.84**

Fig. 1. Total Query Time (seconds) for XBench Queries

From the timing results presented, we conclude that the relational approach is highly scalable to increasing collection sizes, and also provides a significant performance improvement over the tree-based approach. The tree-based approach, on the other hand is increasingly less able to handle queries as the collections get larger. Furthermore, the superior performance of the relational approach shows the potential to exhibit similar performance on very large XML collections.

References

[1] N. Suizo, “Xml propels security intelligence,” *Network World*, August 2006.

[2] D. Florescu and D. Kossman, “Storing and querying XML data using an RDBMS,” *IEEE Data Engineering Bulletin*, vol. 22, no. 3, pp. 27–34, 1999.

[3] W. Meier, “eXist: An open source native XML database,” *Web, Web-Services, and Database Systems. NODe 2002 Web- and Database-Related Workshops*, October 2002, springer LNCS Series, 2593.