Deferred Incremental Refresh of XML Materialized Views*

Hyunchul Kang and JaeGuk Lim**

Dept. of Computer Science and Engineering, Chung-Ang University
Seoul, 156-756, Korea
{hckang,jglim}@dblab.cse.cau.ac.kr

Abstract. The view mechanism can provide the user with an appropriate portion of database through data filtering and integration. In the Web era where information proliferates, the view concept is also useful for XML documents in a Web-based information system. Views are often materialized for query performance, and in that case, their consistency needs to be maintained against the updates of the underlying data. They can be either recomputed or incrementally refreshed by reflecting only the relevant updates. In this paper, we address the issues in deferred incremental refresh of XML materialized views.

1 Introduction

In database systems, the view concept has been a useful and effective mechanism in accessing and controlling data. It is related to many aspects of data management and database design. Among others, one of the most important applications of the view is information filtering and integration, functionality which is getting even more crucial for information engineering in today's Web-based computing environment where vast amount of heterogeneous information proliferates every day.

Since XML emerged as a standard for Web documents, many research issues in XML data management have been investigated. The view concept is also useful for XML data [2][4][5]. For the frequently submitted query against XML documents, its expression can be defined as an XML view for later convenient and efficient reuse. For example, the frequently accessed portions out of huge collection of XML documents in the XML warehouse could be defined as views and possibly maintained as the materialized ones for query performance. Recently, a dynamic large-scale warehouse for XML documents of Web, out of Xyleme project [8] at INRIA, France, which provides sophisticated database-like services like querying, update control, and data integration has already been in commercial service [9].

* This work was supported by grant No. R01-2000-00272 from the Basic Research Program of the Korea Science & Engineering Foundation.

** Current address: R&D center, Tong Yang Systems Corp., Seoul, 138-130, Korea.

A. Banks Pidduck et al. (Eds.): CAISE 2002, LNCS 2348, pp. 742-746, 2002.
© Springer-Verlag Berlin Heidelberg 2002

This paper address the issues involved in management of XML materialized views. When XML views are materialized, their consistency needs to be maintained against the updates of the underlying documents either by recomputing or by incrementally refreshing the outdated portion of the materialization. The latter can be done either immediately after the source document update occurs or in a deferred way. In this paper, we focus on deferred incremental refresh of XML materialized views.

The problem of incremental refresh of materialized views received much attention in relational database systems [3]. The same problem was investigated for the views over semistructured data [1][10][7] in the context of semistructured DBMS. In this paper, we investigate the problem with the XML repository built on top of a relational or an object-relational DBMS instead of the semistructured one. Such an approach deserves thorough investigation because of its pragmatic importance.

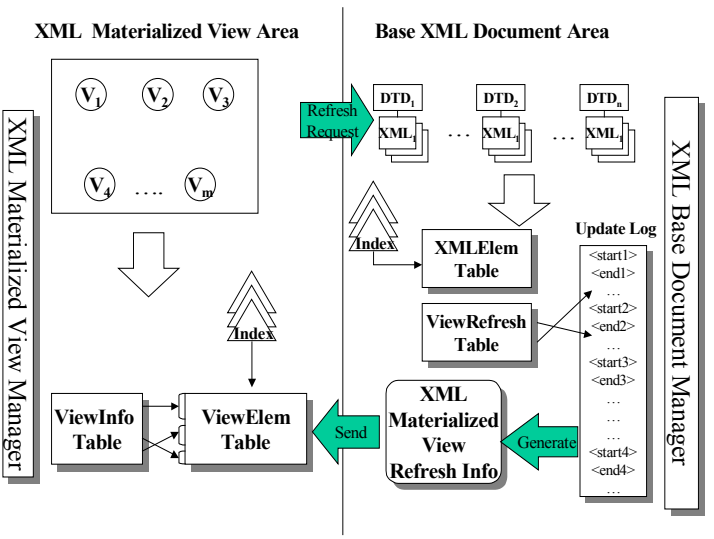

Fig. 1. Framework of XML Materialized View Refresh

2 Framework of XML Materialized View Refresh

Fig. 1 shows a framework of deferred incremental refresh of XML materialized views using a DBMS. The XML repository consists of two areas: the underlying base XML document area and the XML materialized view area. The former is managed by the base document manager, and the latter by the view manager. In the base document area, the DTDs and the XML documents conforming to them are stored. Document indexing is provided for fast access to them. In the view area, the materialized views and the information on the views such as their definition are stored. Indexing is also provided for fast retrieval and refresh of the materialized views. Each update done to the base XML documents is recorded in the update log chronologically for deferred incremental refresh of materialized views. View refresh is done when the view is

requested by an application. That is, the updates are neither immediately nor periodically propagated to the relevant views. Such a materialized view access model is the one employed in [6]. The scenario for retrieval of an XML materialized view is as follows: When view V is requested, the view manager requests the document manager to send it the information necessary for V's refresh. Then, the document manager examines the update log to figure out which updates done to the base documents thus far are relevant to V, generates the view refresh information, and sends it to the view manager. Now the view manager refreshes V as directed by the received view refresh information, and then provides up-to-date V.

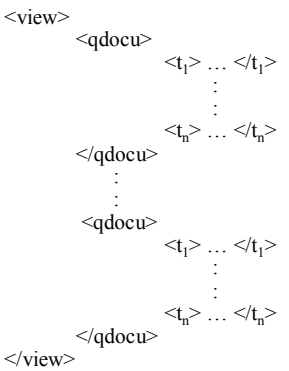

Fig. 2. Template of an XML Materialized View Document

3 Issues in XML Materialized View Refresh

To investigate the XML materialized view, the first thing we need to have is a model of the XML view. One simple and yet practical model is the one whereby an XML view is derived from the base XML documents that conform to the same (and therefore, a single) DTD, and defined by an XML query language expression. The components of XML view definition include the filtering condition, the elements against which the condition is specified, the target elements, and the information for query result restructuring. Complexity of XML view definition allowed directly affects the process of view refresh. Complex views require more work than simpler ones both in checking an update's relevance to them and in generating their refresh information. For deferred incremental refresh of views to be effective, it is desirable to perform both of the above-mentioned tasks with no or least access to the source data. The next thing we need to have is a scheme for representation of XML view materialization. One scheme is to represent the XML materialized view as an XML document. Fig. 2 shows the template of an XML materialized view document. Each element 'qdocu' which stands for 'the base document qualified for the view' is for a base document that satisfies the view's filtering condition, and its subelements 't_i', $i = 1, \ldots, n$, are the target elements of the view retrieved from that particular base document.

Further issues to be resolved based on the model of XML materialized views such as the one described above include the following: First, the database schema for storing not just XML documents but materialized views and other information for their refresh needs to be designed. Secondly, the post optimization in generating the view refresh information with the deferred view refresh policy needs to be considered. Once the view refresh information is generated by scanning the update log, it could be optimized by merging the related refresh information. For example, if an element of a newly inserted document is modified later, then their refresh information can be merged so that the modified value may be inserted instead of the original one. Thirdly, a scheme for efficient logging of updates needs to be devised. Especially, we need to avoid the log records of very large size when inserting a large document or modifying a large element. The logged values of elements are redundantly stored in the corresponding base documents. As such, some referencing mechanism where a log record points to its relevant portion of the base document, is desirable. The penalty for that is on the process of checking the update's relevance to a view, which inevitably requires access to the base documents. Finally and most importantly, the efficient algorithms to check relevance between an update and a view, to generate view refresh information for the relevant pairs, and to reflect the refresh information to the view materialization should be carefully devised. The complexity of these algorithms depends on the granularity of XML document updates as well as on the complexity of view definition allowed.

4 Concluding Remarks

We are currently working on development of an XML repository system that supports XML materialized views using an object-relational DBMS. The performance of the deferred incremental refresh scheme is compared with that of view recomputation. One of the most influencing performance parameters is the amount of logged updates to be examined for deferred incremental refresh. The preliminary results show that for a large collection of XML documents of moderate size and a practical view, the deferred incremental refresh scheme outperforms the recomputation counterpart as long as the proportion of the base XML document updates is less than about 30%.

References

1. S. Abiteboul et al., "Incremental Maintenance for Materialized Views over Semistructured Data," *Proc. Int'l Conf. on VLDB*, 1998, pp. 38-49.
2. S. Abiteboul, "On Views and XML," *Proc. ACM Symp. on Principles of Database System*, 1999, pp. 1-9.
3. A. Gupta and I. Mumick, "Maintenance of Materialized Views: Problems, Techniques, and Applications," *IEEE Data Eng. Bulletin*, Vol. 18, No. 2, Jun. 1995, pp. 3-18.
4. S. Kim and H. Kang, "XML Query Processing Using Materialized Views," *Proc. Int'l Conf. on Internet Computing*, Jun. 2001, pp. 111-117.

5. Y. Papakonstantinou and V. Vianu, "DTD Inference for Views of XML Data," *Proc. of 19th ACM SIGACT-SIGMOD-SIGART Symp. on PODS*, 2000.
6. N. Roussopoulos, "An Incremental Access Method for ViewCache: Concept, Algorithms, and Cost Analysis," *ACM Trans. on Database Systems*, Vol. 16, No. 3, Sep. 1991, pp. 535-563.
7. D. Suciu, "Query Decomposition and View Maintenance for Query Languages for Unstructured Data," *Proc. Int'l Conf. on VLDB*, 1996, pp. 227-238.
8. L. Xyleme, "A Dynamic Warehouse for XML Data of the Web," *IEEE Data Engineering Bulletin*, Vo. 24, No. 2, Sep. 2001, pp. 40-47.
9. Xyleme Home Page, *http://www.xyleme.com*.
10. Y. Zhuge and H. Garcia-Molina, "Graph Structured Views and Their Incremental Maintenance," *Proc. Int'l Conf. on Data Engineering*, 1998, pp. 116-125.