A framework for retrieving and storing differential XML descriptions

Yusuke Nishimura (410095)

School of Information Science,
Japan Advanced Institute of Science and Technology

February 9, 2006

Keywords: XML, differential description, XPath, query evaluation.

We have used XML data for not only data exchange, but also internal data scheme. XML format is more useful for modeling data in variable schema and to contain very long sentences than relational datamodel which consists of static schema.

In some applications, we need to manage huge number of very similar XML data. In such applications, it is desirable to describe the data by describing the common part only once and also describing only the difference for each data. This paper proposes a framework for such description of XML data. In our framework, one can include the whole or some part of a XML data into another XML data, and in that inclusion, one can also modify only part of the data to include in order to customize that data for one's purpose. Our framewrok can reduce volume of data and time of updating the common part and make the difference clear.

We also propose storage and query evaluation schemes for XML data described in our framework. We store our XML data in RDB based on the path-based encoding scheme, and we develop a scheme for efficient evaluation scheme for XPath queries on such storage scheme.

There are three methods of storing the XML data described differential description. The first method is creating index to transform differential descriptions into a complete XML data with interpreting the differntial descriptions when we store it in RDBMS. In this method, we can handle

the differential descriptions as complete ones. However, there's a possibility that it is complex to update the XML data and that the volume of index is huge.

By contrast, the second method is a not creating index. In the method, we need not update an index when we update a XML data. Moreover, we can save volumne of data. However, when we query to RDBMS with SQL, we must interpret the differential description and transform a query which an user describes into one which access the differntial description. Therefore the method consumes more time than the first method.

And we propose the intermediate method between the first method and second one. The method is to create tree-structured indexes which consist of nodes which have an information of connecting differential descriptions. The indexes don't have an usual XML data. They only have an data to connect diffential descriptions. Therefore, The size of index doesn't become huge. Moreover, the information in the nodes can make a query for differential descriptions more simple.

This paper describes the relational data scheme for implementing the three method. In practice, we experiment for measuring volume of data and speed of executing an naive query using a small model which base on this scheme we propose.

In consequence of this experiment, using the first method we retrieve more quickly than the second method as well as the third method. However, we must consume more volume of data than the third method. Moreover, using the first method, it takes an long time to update an XML data in some case.

Therefore, we can use the first method for storing data which we seldom update and storing relatively a small XML data.

It is difficult to use the second method in reality. Because it takes an awful long time to retrieve an XML data by way of differntial descriptions. Moreover, we can not transform all user's query into one which access the differntial description. If there is an recursive description in target data, we cannot create SQL which is able to cover all reference pattern.

We can use the third method for storing data which we frequently update and which we want to save space of an disk. Moreover, we can retrieve differntial descriptions with XPath in adequate time for general require-

ments.

However, we only experiment in small circumstance about some naive XPath queries. We need to acquire more detail result in large data file on another Relational Database Manager System (ex. Oracle, DB2, SQL-Server) about more complex XPath query contains XPath's predication and axies.