Storing XML (with XSD) in SQL Databases: Interplay of Logical and Physical Designs

Surajit Chaudhuri Zhiyuan Chen
Microsoft Research
{surajitc, zhchen}@microsoft.com

Kyuseok Shim
Seoul National University
shim@ee.snu.ac.kr

Yuqing Wu
University of Michigan
yuwu@eecs.umich.edu

XML is becoming the standard for exchanging and querying information across enterprises. Furthermore, much of e-business XML data increasingly relies on accompanying XSD schema specifications (http://www.w3.org/XML/Schema) to ensure semantically meaningful exchanges of information. Languages such as XPath and XQuery have been proposed for querying XML data. One approach towards supporting query over such XML data is that of building native XML storage and query engine [3]. Alternatively, in many scenarios, "shredding" XML data (with its associated XSD specification) into a relational database is an attractive alternative for storage as it can take the full advantage of mature relational database technology. The latter approach requires us to accomplish the following two tasks to ensure efficient execution of XPath queries over XML data: (1) design the *logical* mapping from XML schema to relational schema; (2) select *physical design structures* (i.e., indexes, materialized views, and partitioning) of the relational database where XML is shredded.

Although efficiency of mapping depends on both of the steps above, past work such as [1] exclusively focus on the logical design step. In this paper, we examine the interplay of logical and physical design, and experimentally demonstrate that: (1) solving the logical mapping and the physical design problem independently leads to a suboptimal solution; (2) taking into account the physical design space impacts the space of logical mapping. Specifically, well-known outlining and inlining mapping options [1] are rendered unnecessary because they are functionally subsumed by two physical design options: indexes and vertical partitioning. On the other hand, we identified mapping options that are important to leverage when a XSD specification includes "choice", "optional", and `maxOccurs`. This is because the above constructs imply complex constraints that are difficult to capture solely via physical design in relational databases. For example, an "optional" element (`minOccurs = 0` and `maxOccurs = 1`) in XSD specifies that itself, its subelements, and attributes are either all null or not null. This corresponds to a complex constraint in relational database that specifies whether a set of columns (possibly from different tables) is null at the same time. This constraint is difficult for user to specify in relational databases (although user can easily specify whether a *single* column may have null values or not), and can not be inferred from the relational schema mapped from the XML schema. However, we can exploit this constraint by splitting the table for that element into two tables, one storing those with the optional elements and the other stores those without. As a result, queries only accessing one partitioned table may have better performance.

Our decision to take into account the interplay of logical and physical design for mapping XML documents requires us to solve a difficult search problem as the the combined space of logical and physical design is extremely large. We propose a *search algorithm* that judiciously explores the extreme large combined space of logical and physical design. The algorithm only searches the XSD-specific logical design options and uses heuristics to further prune the search space. We experimentally compare the *quality* (in terms of the time to execute the query workload on resulting design) and *efficiency* (in terms of the search time) of our algorithm with known algorithms as well as a default XSD based mapping and an Edge-Table Mapping [2] that does not use XSD on both real and synthetic data. The results demonstrate the quality and efficiency of our solution is significantly better than previously known techniques.

References

[1] P. Bohannon, J. Freire, P. Roy, and J. Simeon. From XML schema to relations: A cost-based approach to XML storage. In *ICDE*, 2002.

[2] D. Florescu and D. Kossmann. Storing and querying XML data using an RDBMS. *IEEE Data Engineering Bulletin*, 1999.

[3] H. V. Jagadish, S. Al-Khalifa, A. Chapman, L. V. S. Lakshmanan, A. Nierman, S. Paparizos, J. M. Patel, D. Srivastava, N. Wiwatwattana, Y. Wu, and C. Yu. Timber: A native XML database. *VLDB Journal 11(4)*, 2002.