VQBD: Exploring Semistructured Data

Sudarshan S. Chawathe
Computer Science Department
University of Maryland
College Park, Maryland 20742
chaw@cs.umd.edu

Thomas Baby
Computer Science Department
University of Maryland
College Park, Maryland 20742
thomas@cs.umd.edu

Jihwang Yeo
Computer Science Department
University of Maryland
College Park, Maryland 20742
jyeo@cs.umd.edu

The VQBD ("vee-cubed") project addresses the following problem: What is the best way to *explore* an XML document of *unknown structure and content*? We use *data exploration* to denote the interactive task of gathering the information needed to use data for purposes such as generating a report, writing queries, building user interfaces, and writing applications. We focus on XML documents that are too large to browse in their entirety, even with the assistance of pretty-printing software (e.g., multi-megabyte or larger XML documents). In a relational or object database, the schema (e.g., table definitions, class definitions, integrity constraints, and stored procedures) provides some of the information necessary for writing queries and applications. However, the schema is rarely sufficient for these tasks. Typically, one must probe and browse the database to discover data coverage, typical and exceptional values, and other information required to gain a better understanding of the database. In an XML environment, the need for such data exploration is much greater because it is quite likely that the XML data of interest is not accompanied by a schema. Indeed, much XML data is *semistructured*, meaning its structure is irregular, incomplete, and frequently changing. The rapid adoption of XML as a data exchange standard makes this *semistructured data exploration* problem increasingly important. The VQBD system allows the structured exploration of arbitrary XML data. We describe some key features very briefly below; a detailed description appears at `http://www.cs.umd.edu/projects/vqbd/`.

• The subtasks of data exploration, viz., visualization, querying, and browsing, are complementary. For example, when a visualization module (e.g., one that plots cities on a map) has been applied to some query results (e.g., cities with no Starbucks stores), it is possible to refine the query through the visualization interface (e.g., by clicking points on the map). VQBD has a modular design with *plug-in* APIs that allow easy incorporation of additional visualization and querying modules.

• The *level of detail* presented to the user scales smoothly over a wide range. More precisely, the system is able to convey a useful summary of the data in a user-specified number of (graphical) objects. For example, if the display (or user) can only accommodate 20 objects, the system displays 20 objects that effectively summarize the database of, say, 50 thousand objects. These summary objects are only rarely simple reflections of database objects. At a low resolution, VQBD uses summary objects from graphical schemas similar to Data Guides and Graph Schemas.

• There is *no required structure*. The system provides acceptable results for any well-formed XML document. While it is reasonable to expect effectiveness and performance to deteriorate as structure weakens, such deterioration is graceful, not catastrophic. As in semistructured databases, structure is descriptive, not prescriptive.

• Any available *explicit structure* is effectively used. For example, if the XML document is accompanied by DTD or RDF definitions to which it conforms, these definitions are used to provide an appropriate structured browsing interface.

• *Implicit structure* is detected and used. In addition to the explicit structure described by DTD, RDF, and similar methods, a given instance of XML data is likely to contain additional (implicit) structure and patterns. For example, although the DTD governing an XML document may permit address elements that are either strings (#PCDATA) or structured (line1, line2, city, state, zip), all addresses in the current instance may be in the latter format. This fact is used to simplify browsing and printing by always presenting addresses in the structured form. Further, the query-by-example interface is modified to signal that accurate searches of the form `address.zip = 12345` are possible since there is no danger of the ZIP code matching, say, a street number (as would be the case if searches were performed on string-valued address elements).

• It is easy to impose and use structure at run time. For example, an XML document may contain string-valued `name` elements. A user may notice (or know, from out-of-band sources) that all the name strings have a specific format (e.g., last name, first name, initials). The system permits such *run-time structure* to be specified at any time during the exploration process and, once specified, this structure is used in a manner analogous to implicit and explicit structure.

Acknowledgments: Bongwon Suh worked on a preliminary version of VQBD. This work was supported by National Science Foundation grants in the CAREER (IIS-9984296) and ITR (IIS-0081860) programs.

ACM SIGMOD 2001 May 21-24, Santa Barbara, California, USA