Chapter IX
Continuous and Progressive XML Query Processing and its Applications

Stéphane Bressan
National University of Singapore, Singapore

Wee Hyong Tok
National University of Singapore, Singapore

Xue Zhao
National University of Singapore, Singapore

ABSTRACT
Since XML technologies have become a standard for data representation, a great amount of discussion has been generated by the persisting open issues and their possible solutions. In this chapter, the authors consider the design space for XML query processing techniques that can handle ad hoc and continuous XPath or XQuery queries over XML data streams. This chapter presents the state-of-art techniques in continuous and progressive XML query processing. They also discuss several open issues and future trends.

INTRODUCTION
XML (extensible markup language) is now a standard for data dissemination and interchange. While, in most application domains, the amount of available data feeds or data streams, whether sensor or engineered data, is generally increasing, the data in particular is increasingly in XML format. To seize the opportunity created by the availability of such a wealth of network accessible timely data, modern applications need the capability to effectively and efficiently process queries to XML data streams.
From individual stock investors to large hedge fund traders, those watching the stock market are interested in monitoring activities of company stocks and derivatives in the light of other news and data related to companies and their business. An investor, considering both technical and fundamental analyses wants to know both volume and price, and sales and revenue figures of company and industry for the stocks in his portfolio of interest. For the purpose of illustration, we consider the following scenario: An investor poses a query that combines a data stream from the stock market (providing latest volume and price) with a data feed reporting fundamental data data (e.g. updated sales figures). The query is a combination of two streams, the stock market ticker (nyse.xml) and the fundamental data streams (sales.xml) as well as an XML document, the listing in the stock exchange (listing.xml) that provides the mapping of the stock ticker symbol to the company name. The XQuery below returns a set of elements resultTuple, which consists of the company name, ticker symbol, sales, last price, and volume of all the stocks in the exchange.

Reading new blogs is often disorienting as bloggers often assume that their readers are familiar with the news on which they are commenting. A possible solution is to automatically combine blog entries with headlines and provide the links to the related news. Both blogs entries and news are often available as RSS (really simple syndication) or atom feeds. Existing RSS/Atom readers provide basic keyword-based filtering and simple feed merging. Instead of relying on the limited capabilities of existing readers and their interface, and since the feeds are in XML, the desired combination can be expressed as an XQuery offering the full expressive power of a query language. Although RSS and Atom sources are more similar to Web pages being pulled than to feeds or streams pushing data, the latter can be simulated by periodic pulling. In the example at hand, the combination of blog entries and news can be achieved by the Xquery given in Figure 2.

The number and scope of possible applications is limited only by our imagination. Their effective and efficient implementation depends on the availability of algorithms, techniques and tools for the processing of continuous and progressive queries to XML data streams. Unlike the processing of queries to XML repositories, applications processing XML data streams do not have a priori access to the complete data. This makes it difficult to index and organize data. At the same time, since data is transient, only limited memory is available for immediate processing. Since data arrives continuously, these applications need XML query processors that can efficiently

**Figure 1. An XQuery query combining technical and fundamental data from live feeds for market monitoring**

```xml
( for $s in doc("nyse.xml")//symbolTuple
    for $t in doc("listing.xml")//company
    for $q in doc("sales.xml")//sales
        where $s/symbol = $t/symbol and $q/company = $t/name
    return <resultTuple>
        {($q/company), ($s/symbol), ($s/sales),
         ($q/lastSale), ($q/shareVolume)}
    </resultTuple>
)```