Publishing & Updating XML with Attribute Transformation Grammars
Not just disposing your data in XML

Wenfei Fan, Byron Choi and Xibei Jia
http://www.lfcs.inf.ed.ac.uk/research/database/dbs.html

Abstract: This poster provides an overview of a new system for schema-directed publishing of relational data in XML, based on the novel concept of Attribute Transformation Grammars (ATGs). A salient feature of this system is that target schema (DTD) conformance is automatically guaranteed. To cope with data that evolves over time, the system also supports an incremental algorithm for maintaining XML data produced by ATGs. The system is the only one that is capable of efficiently publishing and maintaining XML views with respect to recursive and disjunctive DTDs.

Schema-Directed Publishing

XML has become the prime standard for data exchange on the Web. To exchange data currently residing in databases, one needs to publish it in XML, i.e., to extract data from the database and transform the data into an XML format. In practice, XML publishing is scheme-directed: the published XML data must conform to a predefined (possibly recursive) schema.

Figure 1. Schema-directed XML publishing & maintenance

We propose the first framework for schema-directed publishing based on the novel notion of Attribute Transformation Grammars (ATGs) [2]. An running example taken from GO (Gene Ontology):

• Some members of the biologist community own gene ontology data (with a source schema R);
• Source relational schema R:
  primary terms ( genID, name, updated)
  term ( termID, name, updated)
  ancestor ( parentID, childID)
  main ( proteinID, name, source)

• the community agrees on a recursive target schema (DTD), D0. Recursions due to term;

Target DTD D0:

• ATGs:
  • schema-directed: automatically guarantees the conformance to D0;
  • stepwise: provide guidance on how to define an XML view that type checks;
  • easy to learn: one can “program” in ATG with basic knowledge of SQL and DTDs.

Incremental Updates

The underlying source data often changes and evolves. To avoid full recreation of the XML views, we propose an efficient incremental algorithm [3] for maintaining the XML views:

• the algorithm minimizes unnecessary recomputations;
• the performance is proportional to the size of the updates instead of the view;
• the updated view is guaranteed to conform to the target schema.

When (group) updates on the underlying data sources are detected, the algorithm maintains the XML view by performing the following steps:

• translate the deletions to “cuts” of subtrees in the view (note that a tuple can correspond to multiple subtrees – a “cut” may affect multiple places in the view);
• locate the places in the view related to the insertions. Mark them as “buds.” “Grow” the buds (again, an inserted tuple may affect multiple places in the view).

The Prata System

The current Prata system (Fig. 2) consists of modules facilitating the following tasks:

1. view and edit the source and target schemas;
2. publish and browse the XML view;
3. issue relational updates and propagate the updates to the published XML view.

Figure 2. System Architecture

Coming Attractions

The following features are being incorporated into the Prata system. Stay tuned.

• a security interface for XML data [6];
• schema-directed integration [1];
• schema-directed XML to XML transformations [8];
• multiple, distributed, heterogeneous sources;
• automatic validation of integrity constraints [4, 5, 7].

References