SMOQE: A System for Providing Secure Access to XML

Wenfei Fan, Floris Geerts, Xibei Jia, Anastasios Kementsietsidis

The University of Edinburgh, Digital Curation Centre

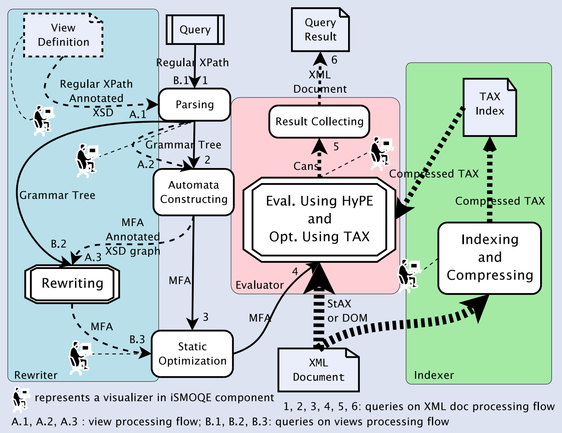
Abstract: XML views have been widely used to enforce access control, support data integration, and speed up query answering. In many applications, e.g., XML security enforcement, it is prohibitively expensive to materialize and maintain a large number of views. Therefore, views are necessarily virtual. This poster presents SMOQE, the first system to provide efficient support for answering queries over virtual and possibly recursively defined XML views.

Introduction

For all the reasons that views are essential to traditional databases, XML views are also important for XML data.

We have developed the Secure MOdular Query Engine (SMOQE) [9] for facilitating the specification of XML views and answering of XML queries on virtual views. The main features of SMOQE are the following.

- SMOQE supports XML views defined by annotating an XML schema with Regular XPath queries. It supports recursively defined schemas (and thus views).
- SMOQE is able to rewrite any Regular XPath query Q posed by users on a virtual view V to an equivalent Regular XPath query Q' on the underlying document T.
- SMOQE encompasses a query engine for Regular XPath queries, implementing an efficient evaluation algorithm and a novel indexing structure.



The SMOQE *Architecture*

System Architecture

As shown above, SMOQE consists of four modules:

(a) *i*SMOQE, a visual tool through which a user can define XML views, inspect the query rewriting and evaluation, and browse results (a small icon is used to indicate all the system components accessible through *i*SMOQE);

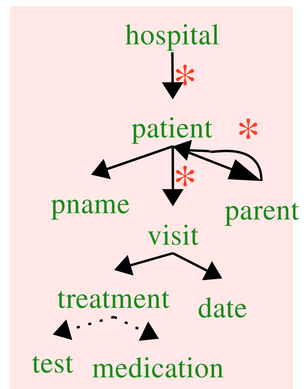
(b) a query rewriter (indicated by a blue box at the left of the figure) for translating user Regular XPath queries posed on XML views to equivalent Regular XPath queries on the underlying document;

(c) a query evaluator (indicated by a pink box in the middle of the figure) for processing Regular XPath queries;

(d) an indexer (indicated by a green box at the right of the figure), which is used by the evaluator to build indexes and optimize queries.

Security Views

An *access specification* S is an extension of a document DTD D by associating security annotations with productions of D [8]. Here is an example:

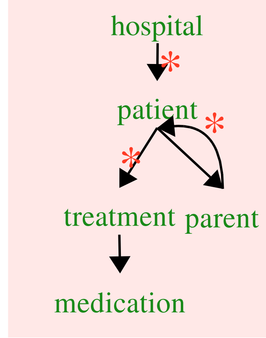


hospital → patient*
ann(hospital,patient) = [visit/treatment/medication = 'autism']
patient → pname, visit*, parent*
ann(patient, pname) = N
ann(patient, visit) = N
parent → patient
visit → treament, date
ann(visit, treatment) = [medication]
treatment → test + medication
ann(treatment, test) = N

document DTD D_0 access specification S_0

A *security view specification* σ could be either manually defined by security administrators or automatically derived from access specifications. Moreover, a view DTD could be provided to help users posing queries by the view derivation algorithm in the latter case:

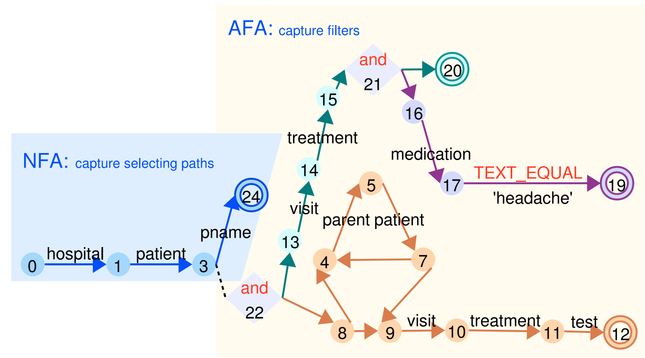
hospital → patient*
σ_0*(hospital, patient) = patient[visit/treatment/medication = 'autism']*
patient → treatment*, parent*
σ_0*(patient, treatment) = visit/treatment[medication]*
σ_0*(patient, parent) = parent*
parent → patient
σ_0*(parent, patient) = patient*
treatment → medication
σ_0*(treatment, medication) = medication*



view specification σ_0 derived from S_0 view DTD D_V

Regular XPath Evaluation

While it is always possible to rewrite a Regular XPath query Q on a view to an equivalent query Q' on the underlying document, the size of Q', if directly represented as Regular XPath expressions, may be exponential in the size of Q [10]. The SMOQE rewriter overcomes this challenge by employing an automaton characterization of Q', denoted by MFA (*mixed finite state automaton*) [10], which is *linear* in the size of Q.



For example, the figure above depicts the MFA $\mathcal{M}_0$ characterizing the Regular XPath query:

Q_0 = hospital/patient[(parent/patient)*/visit/treatment/test/ and visit/treatment[medication/text()="headache"]]/pname

The SMOQE evaluator implements a novel algorithm, referred to as HyPE (Hybrid Pass Evaluation) [10], for processing Regular XPath queries represented by MFA's. Fig. 3 shows the evaluation of the MFA $\mathcal{M}_0$ given earlier on an XML document:

A unique feature of HyPE is that it needs a single top-down depth-first traversal of the XML tree, during which HyPE both evaluates predicates of the input query (equivalently, AFA's of the MFA) and identifies potential answer nodes (by evaluating the NFA of the MFA). The potential answer nodes are collected and stored in an auxiliary structure, referred to as Cans (candidate answers), which is often much smaller than the XML document tree. A pass over Cans is needed to retrieve the real result nodes.

The Big Picture

Data Cleaning [6, 5] When overlapping or redundant information from multiple sources is integrated, inconsistencies in the data may emerge as violations of integrity constraints on the integrated data. The constraint repair problem attempts to find "low cost" changes that, when applied, will cause the constraints to be satisfied.

XML Publishing [2, 3, 7] A common paradigm for data exchange on the Web is to first convert data to XML, and then send the XML data over the network to another party. We consider a systematic DTD-directed approach to publishing data while guaranteeing schema-conformance. Effective ways to propagate the updates from the data sources to the target XML document (XML views) without violating the predefined schema and with minimal recomputation is also investigated.

Data Integration [1] While XML publishing is nontrivial, schema-directed XML integration is much harder. It typically needs to extract data from a collection of distributed, heterogeneous data sources, and construct an XML document that conforms to a predefined schema.

Schema Embedding [4] A fundamental concern of information integration is the ability to embed instances of one or more source schemas in an instance of a target schema so that the information in the source instances is preserved. This calls for a notion of schema embedding that enables a source schema to be effectively matched to (or, embedded in) a target schema with larger "information capacity".

References

[1] M. Benedikt, C. Y. Chan, W. Fan, J. Freire, and R. Rastogi. Capturing both types and constraints in data integration. In *SIGMOD*, 2003.

[2] M. Benedikt, C. Y. Chan, W. Fan, R. Rastogi, S. Zheng, and A. Zhou. DTD-directed publishing with attribute translation grammars. In *VLDB*, 2002.

[3] P. Bohannon, B. Choi, and W. Fan. Incremental evaluation of schema-directed XML publishing. In *SIGMOD*, 2004.

[4] P. Bohannon, W. Fan, M. Flaster, and P. P. S. Narayan. Information preserving XML schema embedding. In *VLDB*, 2005.

[5] P. Bohannon, W. Fan, M. Flaster, and R. Rastogi. A cost-based model and effective heuristic for repairing constraints by value modification. In *SIGMOD*, 2005.

[6] P. Bohannon, W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis. Conditional functional dependencies for data cleaning. In *ICDE*, 2007.

[7] B. Choi, W. Fan, X. Jia, and A. Kasprzyk. A uniform system for publishing and maintaining XML. In *VLDB*, 2004. Demo.

[8] W. Fan, C. Y. Chan, and M. Garofalakis. Secure XML querying with security views. In *SIGMOD*, 2004.

[9] W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis. SMOQE: A system for providing secure access to XML. In *VLDB*, 2006. Demo.

[10] W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis. Rewriting regular XPath queries on XML views. In *ICDE*, 2007.

School of informatics