Tree Logical Classes for Efficient Evaluation of XQuery

Stelios Paparizos, Yuqing Wu, H.V. Jagadish, Laks V.S. Lakshmanan

Presented by:
Todd J. Green
University of Pennsylvania
October 7, 2004
Running Example

Here is an XQuery query.

FOR $p$ IN document(‘‘auction.xml’’)//person
FOR $o$ IN document(‘‘auction.xml’’)//open_auction
WHERE count($o/bidder) > 5 AND $p/age > 25
    AND $p/@id = $o/bidder//@person
RETURN
    <person name={$p/name/text()}> $o/bidder </person>

We all can see what the query means. But how do you actually evaluate it? That is the focus of this paper.
Evaluating XQuery

Of course, this is not the first paper ever about evaluating XQuery. There have been many. The approaches fall into three categories:

1. Mapping to relational

2. Native navigational-based (Galax)

3. Native algebraic-based (Tamino, Xyleme, …)

This paper is one of those which take the third approach. Moreover, theirs is “tree-based” rather than “node-based”. This means something like, intermediate results are collections of trees rather than collections of vertices.
Matching

Before we try anything fancy, let’s just consider the fundamental operation of matching. There are two crucial concepts used in matching: the pattern tree and the witness tree. The exposition in the paper is not very clear (witness tree is not even defined at all), yet these are crucial concepts, so we will spend a little time here.
Annotated Pattern Tree

First some intuition. We use something called a “pattern tree” to perform projection. A “match” of the pattern tree on the “database” $D$ yields an XML tree $T$ which we call the “witness tree”. It “witnesses” the match by virtue of having a mapping $f$ back to the pattern $P$ and a mapping $g$ forward to the database $D$, like so:

$$P \xleftarrow{f} T \xrightarrow{g} D$$

The pattern $P$ imposes rules on the allowed mappings $f$ and $g$ to enforce things like cardinality (‘*’, ‘?’, etc) and ancestry relationships (‘/’ versus ‘//’).

OK, now on to the details.
Annotated Pattern Tree

An annotated pattern tree $P$ is a quintuple $P = \langle V, E, m, p, r \rangle$, where:

- $(V, E)$ is a tree;
- $m : E \rightarrow \{\ast, ?, -, +\}$ is a matching specification function mapping each edge to a cardinality constraint;
- $p$ is a mapping that associates with each vertex $v$ in $V$ a predicate $p_v$;
- $r : E \rightarrow \{/, //\}$ is a function specifying for each edge $(u, v)$ in the pattern the desired structural relationship between $u$ and $v$ (parent-child or ancestor-descendant).
Witness Tree

Let $D$ be a set of trees representing the database, and let $P = \langle V, E, m, p, r \rangle$ be an annotated pattern tree. We say that a tree $W$ is a witness tree for $D$ and $P$ if there exists a mapping $f : W \rightarrow D$ such that

- $f$ is a homomorphism, that is,
  1. $f(\text{root}(W)) = \text{root}(P)$,
  2. for all $(u, v) \in W$ we have $(f(u), f(v)) \in P$.

- $f$ satisfies the cardinality constraints of $P$, that is, for all $u \in P$, we have:
  1. $|f^{-1}(u)| = 0$ implies $m(u) = '?' \lor m(u) = '*'$,
  2. $|f^{-1}(u)| > 1$ implies $m(u) \neq '-' \land m(u) \neq '?'$. 
and there exists a mapping $g : W \rightarrow D$ such that

- $g$ is injective (one-to-one)
- $g$ satisfies the ancestry constraint of $P$, that is, for all $(u, v) \in W$, we have
  1. $r(f(u), f(v)) = '/'$ implies $(g(u), g(v)) \in D$,
  2. $r(f(u), f(v)) = '//'$ implies there exists a path $t$ from $g(u)$ down to $g(v)$ in $D$.
- $g$ satisfies the predicate constraint of $P$, that is, for all $u \in W$, $p_{f(u)}(g(u)) = true$

As a corollary, the composition $h = g \circ f^{-1} : P \rightarrow 2^D$ is what the paper calls a match.
Example

Figure 4: A sample match for an Annotated Pattern Tree. Any edge without an explicit annotation implies the default “-”. The double-edged lines represent an ancestor-descendant relationship. Note how the APT addresses heterogeneity on both dimensions (height and width) using variations of annotated edges.
Logical Classes

OK, we’re almost ready to look at the algebra itself. But first, we need one more key concept, that of the logical class. Intuitively, a logical class is just a set of nodes in the database that get matched by a particular node in the pattern tree. More precisely, let $P$ be a pattern tree and let $\mathcal{W}$ denote the set of witness trees for $P$ on some database $D$. Then we define the logical class of a vertex $v$ in $P$ to be

$$LC(v) = \{u \mid \exists W \in \mathcal{W} \text{ s.t. } u \in W \land f(u) = v\}$$

For each logical class $LC(v)$ we additionally associate a unique label $LCL_v$ identifying the class.
The Tree Logical Algebra

With the required machinery under our belts, we are ready to look at the algebra itself. There are seven algebraic operators in total. They all map sets of trees to sets of trees. The first six are unary:

1. **Select** $\sigma_P : S \rightarrow S$
2. **Project** $\pi_{nl} : S \rightarrow S$
3. **Filter** $\phi_{LCL_{f,p,m}} : S \rightarrow S$
4. **Duplicate Elimination** $\delta_{nl,ci} : S \rightarrow S$
5. **Aggregate Function** $\alpha_{fname,LCL_a,LCL_b} : S \rightarrow S$
6. **Construct** $\kappa_c : S \rightarrow S$

There is also one binary infix operator:

7. **Join** $\bowtie_{P,p} : S \times S \rightarrow S$
Let’s go over a few of these.
Select

1. Select $\sigma_P : S \rightarrow S$

$P$ is an annotated pattern tree. For each tree in the input set $S$, the operator performs a pattern tree match using $P$. The output is the entire set of the matching witness trees for all input trees.
Project

2. **Project** $\pi_{nl} : S \rightarrow S$

Here, $nl$ is a list of logical class labels. Roughly speaking, for each input tree, retain only the nodes in the logical classes identified by $nl$. 
3. **Filter** $\phi_{LCL_f,p,m} : S \rightarrow S$

Given a set of trees $S$, a predicate $p$, and a mode flag $m$, the meaning of $\phi_{LCL_f,p,m}(S)$ is the following: output only the trees in $S$ that satisfy the predicate $p$ (a) for every node $u \in S$ in logical class $LC_f$ (if $m = \forall$), or (b) for some node $u \in S$ in logical class $LC_f$ (if $m = \exists$). Actually, there are additional modes too, like “exactly one” and “first element.”
Join

7. **Join** $\Join_{P,p} : S \times S \rightarrow S$

Let $S_1$ and $S_2$ be two sets of trees. Let $P$ be an annotated pattern tree. Let $p$ be a predicate. Question: what does $S_1 \Join_{P,p} S_2$ mean?
Running Example

Now that we have gotten the flavor of the algebra, let’s revisit the XQuery example from earlier:

FOR $p$ IN document(‘‘auction.xml’’)//person
FOR $o$ IN document(‘‘auction.xml’’)//open_auction
WHERE count($o/bidder) > 5 AND $p/age > 25
    AND $p/@id = $o/bidder//@person
RETURN
    <person name={$p/name/text()}>$o/bidder </person>
Running Example

Here is how the TLA query plan looks.
Optimizations

Performing optimizations means eliminating or avoiding unnecessary work. Here, the expensive operation that we want to avoid doing too much is tree pattern matching. To that end, the authors have developed various techniques to “reuse” pattern tree matches. These are physical algebraic operators $FL$ (flatten) and $SH/IL$ (shadow and illuminate).
Flatten

If the following figure appears blurry, please consider scheduling an appointment with your eye doctor.

Figure 9: Flatten Operator
Shadow/Illuminate

Another illuminating example.

Figure 11: Comparing operators: Flatten vs. Shadow
The End