XML-Relational Mapping

CPS 116

Introduction to Database Systems

Announcements (October 28)

- Homework #2 has been graded
- Homework #3 out last Thursday; due next Thursday
- Project milestone #2 due in 2 weeks
  - Check your email for feedback

Approaches to XML processing

- Text files (!)
- Specialized XML DBMS
  - Lore (Stanford), Strudel (AT&T), Timber (Michigan), MonetDB/XQuery (CWI, Netherlands), Tamino (Software AG), eXist, Sedna, Apache XIndice, XML:DB API initiative…
  - Still a long way to go
- Object-oriented DBMS
  - ObjectStore, ozone, …
  - Not as mature as relational DBMS
- Relational (and object-relational) DBMS
  - Middleware and/or object-relational extensions

Mapping XML to relational

- Store XML in a CLOB (Character Large OBject) column
  - Simple, compact
  - Full-text indexing can help (often provided by DBMS vendors as object-relational "extensions")
- Alternatives?
  - Schema-oblivious mapping:
    well-formed XML → generic relational schema
    - Node/edge-based mapping for graphs
    - Interval-based mapping for trees
    - Path-based mapping for trees
  - Schema-aware mapping:
    valid XML → special relational schema based on DTD

Node/edge-based: schema

- *Element*(*eid*, *tag*)
- *Attribute*(*eid*, *attrName*, *attrValue*) Key:
  - Attribute order does not matter
- *ElementChild*(*eid*, *pos*, *child*) Keys:
  - *pos* specifies the ordering of children
  - *child* references either *Element*(*eid*) or *Text*(*tid*)
- *Text*(*tid*, *value*)
  - *tid* cannot be the same as any *eid*
- Need to "invent" lots of *id*'s
- Need indexes for efficiency, e.g., *Element*(*tag*), *Text*(*value*)

Node/edge-based: example

```
<bibliography>
  <book ISBN="ISBN-10" price="80.00">
    <title>Foundations of Databases</title>
    <author>Abiteboul</author>
    <author>Hull</author>
    <author>Vianu</author>
    <publisher>Addison Wesley</publisher>
    <year>1995</year>
  </book>
</bibliography>
```

Element

eid	tag
e0	bibliography
e1	book
e2	title
e3	author
e4	author
e5	author
e6	publisher
e7	year

ElementChild

eid	pos	child
e0	1	e1
e1	1	e2
e1	2	e3
e1	3	e4
e1	4	e5
e1	5	e6
e1	6	e7
e2	1	t0
e3	1	t1
e4	1	t2
e5	1	t3
e6	1	t4
e7	1	t5

Attribute

eid	attrName	attrValue
e1	ISBN	ISBN-10
e1	price	80

Text

tid	value
t0	Foundations of Databases
t1	Abiteboul
t2	Hull
t3	Vianu
t4	Addison Wesley
t5	1995

Node/edge-based: simple paths

- //title
  - SELECT eid FROM Element WHERE tag = 'title';
- //section/title
  - SELECT e2.eid
    FROM Element e1, ElementChild c, Element e2
    WHERE e1.tag = 'section'
    AND e2.tag = 'title'
    AND e1.eid = c.eid
    AND c.child = e2.eid;
- Path expression becomes

Node/edge-based: more complex paths

- //bibliography/book[author="Abiteboul"]/@price
  - SELECT a.attrValue
    FROM Element e1, ElementChild c1,
         Element e2, Attribute a
    WHERE e1.tag = 'bibliography'
    AND e1.eid = c1.eid AND c1.child = e2.eid
    AND e2.tag = 'book'
    AND EXISTS (SELECT * FROM ElementChild c2,
                        Element e3, ElementChild c3, Text t
                WHERE e2.eid = c2.eid AND c2.child = e3.eid
                AND e3.tag = "author"
                AND e2.eid = c3.eid AND c3.child = t.tid
                AND t.value = "Abiteboul")
    AND e2.eid = a.eid
    AND a.attrName = "price";

Node/edge-based: descendent-or-self

- //book//title
  - Requires SQL3 recursion
  - WITH ReachableFromBook(id) AS
    ((SELECT eid FROM Element WHERE tag = 'book')
     UNION ALL
     (SELECT c.child
      FROM ReachableFromBook r, ElementChild c
      WHERE r.eid = c.eid))
    SELECT eid
    FROM Element
    WHERE eid IN (SELECT * FROM ReachableFromBook)
    AND tag = 'title';