XML Query Languages (2)

Prof. Weining Zhang

XML-QL

- Proposed to W3C in 1998
- Based on SQL and a semistructured data model
- Specify patterns as well as construction
- Support both an ordered and an unordered data model
- Support regular path expressions
- Very simple syntax

A Simple Query

- Find all authors of books whose publisher is Addison-Wesley in the XML document at www.a.b.c/bib.xml
WHERE <book>
  <publisher>
    <name>Addison-Wesley</name>
  </publisher>
  <title>$t</title>
  <author>$a</author>
</book> IN "www.a.b.c/bib.xml"
CONSTRUCT $a

Another Look

- With simplified syntax
WHERE <book>
  <publisher>
    <name>Addison-Wesley</name>
  </publisher>
  <title>$t</title>
  <author>$a</author>
</book> IN "www.a.b.c/bib.xml"
CONSTRUCT $a

Construct New XML Data

- List author & title pairs s.t., the author publishes a book in that title
WHERE <book>
  <publisher>
    <name>Addison-Wesley</name>
  </publisher>
  <title>$t</title>
  <author>$a</author>
</book> IN "www.a.b.c/bib.xml"
CONSTRUCT <result>
  <author>$a</author>
  <title>$t</title>
</result>

A nested Query

- List book titles & group authors under each title
WHERE <book>
  <title>$t</title>
  <publisher>
    <name>Addison-Wesley</name>
  </publisher>
</book> CONTENT AS $p "www.a.b.c/bib.xml"
CONSTRUCT <result>
  <title>$t</title>
  WHERE <author>$a</author> IN $p
  CONSTRUCT <author>$a</author>
</result>
A Join Query

- Retrieve all articles that have at least one author who has also written a book since 1995

```
WHERE <article><author>
  <firstname> $f </firstname>
  <lastname> $l </lastname>
</article><author>
CONTENT AS $a IN "www.a.b.c/bib.xml",
<book year>=$y</book year>,<author>
<firstname> $f </firstname>
<lastname> $l </lastname>
</author>
CONTENT AS $a IN "www.a.b.c/bib.xml",
$y > 1995
CONSTRUCT <article> $a </article>
```

The XML-QL Data Model

- Two models: unordered & ordered
- Model XML data as a labeled graph
  - Each node has a unique OID
  - Each arc is labeled with an element tag
  - Each internal node is labeled with sets of attribute-value pairs
  - Each leaf node is labeled with values (strings),
  - There is a distinguished root node
- IDREF & IDREFS attributes are modeled by arcs labeled with the attribute names
- Each leaf node has one value

Tag Variable

- Finds all publications published in 1995 in which Smith is either an author, or an editor

```
WHERE <p>
  <title> $t </title>
  <year>1995</year>
  <$e> Smith </$e>
</p>
IN "www.a.b.c/bib.xml",
$e IN { author, editor }
CONSTRUCT <p>
  <title> $t </title>
  <$e> Smith </$e>
</p>
```

Regular Path Expression

- Used to specify arbitrary paths & cyclic structure
  - <A*> matches any number of nested <A> elements
  - <*> matches any number of any elements
  - Also, alternation (|), concatenation (.), and Kleene-star (*) operators
- Ex:
  WHERE
  <part+.part+component.piece>[part+]<$/p>
IN "www.a.b.c/parts.xml"
CONSTRUCT <results> $r </results>
```

Skolem Function

- Used to generate a unique OID for an element and to group sub-elements of the element during construction of new structure

```
WHERE <$p> <author> <firstname> $f </firstname>
  <lastname> $l </lastname>
</author>
IN "www.a.b.c/bib.xml",
CONSTRUCT <author ID=PersonID($fn, $ln)>
  <firstname> $f </firstname>
  <lastname> $l </lastname>
</author>
```

Blocks of Queries

- Each subblock generates a structure and the skolem function matches structures with the same OID

```
{ WHERE <person> <name> </name> ELEMENT AS $n
  <$sn> $ssn </$sn>
</person>
IN "www.a.b.c/data.xml"
CONSTRUCT <result ID=SSNID($ssn)> $n </result>
{ WHERE <taxpayer> <ssn> $ssn </ssn>
  <income> </income>
  ELEMENT AS $i
</taxpayer>
IN "www.irs.gov/taxpayers.xml"
CONSTRUCT <result ID=SSNID($ssn)> $i </result>
}
```
Other Features

- Ordered variables
- Order-By clause
- Embedding XML-QL queries in data
- User-defined functions

XML-GL

- A graphical query language for XML, 1998
- Based on a general logic- & graph-based language for semistructured data model
- Intuitive to represent the hierarchical nature of XML
- Suitable to interpret querying as a subgraph extraction and restructuring as a graph creation
- Easy to use by non-technical persons

XML-GL Data Model

- XML-GDM (Graphical Data Model)
  - Graphical representation of DTDs and Documents
- Building blocks
  - XML Elements
    - Depicted as rectangles
    - Correspond to XML items with no direct value
  - XML Properties
    - Depicted as circles (content nodes as white circles and attribute nodes as black circles)
    - Correspond to XML printable values (PCDATA)
  - Relationships
    - Depicted as directed arcs
    - Correspond to containment or reference

DTDs and XML-GDM

Features of XML-GDM

- Cardinality constraints (0:1, 0:N, 1:1, 1:N) are applied to arcs
  - Default: 1:1
- Ordering of sub-elements (if required):
  - The first arc has a small trait
  - Then the arcs are counterclockwise ordered
- Most XML attributes are represented as properties
- Those having an IDREF or IDREFS domain, are depicted as arrows
- Alternatives are represented by xor arcs

Advanced XML-GDM graph
The four tasks of (XML) queries

- **EXTRACT**
  - Scope of the query, where the information originates
    - SQL: from clause
- **MATCH**
  - Properties of the desired result
    - SQL: where clause
- **CLIP**
  - Information to be retained
    - SQL: select clause
- **CONSTRUCT**
  - How information should be restructured
    - SQL: create view

An Extract-Clip query /1

- Return all the person elements
  - Only the person elements appear in the result (with their attributes)

An Extract-Clip query /2

- Return all the person elements
  - The person elements appear in the result with their first level sub-elements

An Extract-Clip query /3

- Return all the person elements
  - The person elements appear in the result with their sub-elements at any level

Structure of an XML-GL query

- Left graph:
  - Extract and Match part
  - Adorned subset of the original DTD
- Right graph:
  - Clip and Construct part
  - DTD of the result
- L-R bindings are the mechanism to pursue correspondences of nodes in the left and right graphs; bindings can be explicit or implicit.

Extract-Match-Clip query

- Find orders containing the book titled “Introduction to XML” shipped to Los Angeles. Present the order with its items and its shipping information.
Ext-Match-Clip query with join
- Find all books written by an author with the same last name of a person whose first name starts with ‘S’

Semantics of XML-GL queries
- Informal description
  - All the combinations of XML elements appearing in the left side are considered
  - Those not satisfying the matching conditions are discarded
  - The remaining combinations are passed one by one to the right side
- Similar to SQL semantics
- Operational semantics formally defined

Constructing new elements /1
- Three different constructors
  - Constructed element (each retrieved item is wrapped by a new element)
  - List Node (all retrieved items are listed as subelements of a common new element)
  - Index Node (items are wrapped according to some criteria)

Constructing new elements /2
- Build a result element for each person living in Los Angeles

Constructing new elements /3
- Build a result element for all persons living in Los Angeles

Constructing new elements /4
- Group all persons living in Los Angeles by their lastname, each group in a result element
Other Features

- Allow expression of computations
- ...