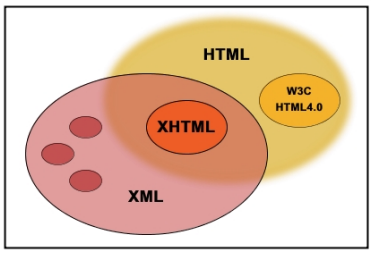
Lecture 8
More XML Awesome



(Sorry, this is as good as XML-related images get.)
(Slides due to Dan Suciu via R. Ramakrishnan)

Quick Review

- XML popular for data exchange
- Remember what XML can model:
  - Missing or additional attributes
  - Multiple attributes
  - Data with irregular structure
  - Good for Web, medical, bio data
- DTDs are kinda like schemas
  - A BNF grammar, constrains elt structure & content
  - Defines entities
- Today: How to use XML as a DB
  - How to think about XML queries
  - How to express queries
  - How to store and process XML data

2

First, more on DTDs

- DTDs are good for docs, have problems when applied to data
  - Elt name, type assoc is global (no scoping)
  - No support for data types, so can't do data validation
    - I was wrong in last lecture!
- You can do this:
  - `<!ATTLIST ARTICLE art-id ID #REQUIRED>`
  - `<!ATTLIST ARTICLE cites IDREF #IMPLIED>`
  - `<article art-id="1">…</article>`
  - `<article cites="1">…</article>`
- But:
  - Only one, and no composite ids
  - No foreign keys (keys to keys)
  - No constraints on IDREF (can ref anything!)

3

XML Schema

- Elts are typeable
- Primitive data types (ints, strs, dates)
- User-defined hierarchical types
  - So can be reused by multiple elts
- Inheritance
- Foreign keys
- Constraints possible on values and on the elt-type of references

4

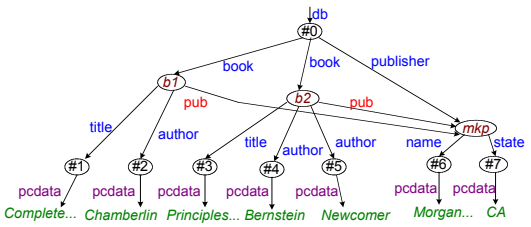
XML Schema

```
<schema version="1.0"
   xmlns="http://www.w3.org/1999/XMLSchema">
<element name="author" type="string" />
<element name="date" type = "date" />
<element name="abstract">
  <type>
    …
  </type>
</element>
<element name="paper">
  <type>
    <attribute name="keywords" type="string"/>
    <element ref="author" minOccurs="0" maxOccurs="*"
   />
    <element ref="date" />
    <element ref="abstract" minOccurs="0" maxOccurs="1"
   />
    <element ref="body" />
  </type>
</element>
</schema>
```

5

XML Data Model

- Sometimes called a "graph data model"



6

XML Query Data Model

- If an XML document is a database, what is an XML *query*?
- "XQuery 1.0 and XPath 2.0 Data Model"
- An idealized version of the XML data we will process

1. Data is tree, in which each node is one of:
   - DocNode
   - ElemNode, ValueNode, AttrNode
   - NSNode, PINode, CommentNode, InfoItemNode, RefNode

7

XML Query Data Model

- elemNode: (QNameValue,
  {AttrNode},
  [ElemNode | ValueNode])
  ⇒ **ElemNode**
- "Give me a tag, a set of attributes, a list of elements/values, and I will return an ElemNode object"

8

ElemNode

Example:

```
<book price = "55"
      currency = "USD">
  <title> Foundations … </title>
  <author> Abiteboul </author>
  <author> Hull </author>
  <author> Vianu </author>
  <year> 1995 </year>
</book>
```

```
book1= elemNode(book,
  {price2, currency3},
  [title4,
   author5,
   author6,
   author7,
   year8])
price2 = attrNode(…)   /* next */
currency3 = attrNode(…)
title4 = elemNode(title, string9)
…
```

9

AttrNode

- attrNode: (QNameValue, ValueNode)
  ⇒ **AttrNode**
- "Give me a name and a ValueNode, and I will return an AttrNode object"

10

AttrNode

Example:

```
<book price = "55"
      currency = "USD">
  <title> Foundations … </title>
  <author> Abiteboul </author>
  <author> Hull </author>
  <author> Vianu </author>
  <year> 1995 </year>
</book>
```

```
price2 = attrNode(price,string10)
string10 = valueNode(…)  /* next */
currency3 = attrNode(currency,
                     string11)
string11 = valueNode(…)
```

11

ValueNode

- ValueNode = StringValue |
  BoolValue |
  FloatValue | …
- stringValue: string ⇒ StringValue
- boolValue: boolean ⇒ BoolValue
- floatValue: float ⇒ FloatValue

12

XQuery

- XQuery 1.0 is the SQL of XML
- XPath 2.0, used to define XSLT templates, is a subset of Xquery
- Currently just extracts data from XML documents
  - Updates are in the works
- Everything's a function (unlike SQL)
- Not in XML!
  - Nice!

13

Remember…

- Path expressions in XSLT? Part of Xquery
- Basic: /bib/paper/author
- Also:
  - //bib
  - /bib/*
  - /bib/paper[2]/author[1]
  - paper[author/lastname="Vianu"]
  - paper[@author="Vianu"]
  - /bib/(paper|book)/title

14

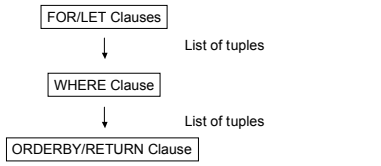
More XPath

- **Axes** define node relationships
  - Ancestor
  - Attribute
  - Child
  - Descendant
  - Descendant-or-self
  - Etc.
- Implicit child: axis in each path step
  - /child::bib/descendant::author
- /step0/step1/step2…
  - Step = axis:nodetest[predicate]

15

XQuery

- FOR-LET-WHERE-ORDER-RETURN
  - FLWOR! Or "flower" queries



16

FOR..LET

- FOR $x in expr
  - Binds $x to each value in the list 'expr'
- LET $x = expr
  - Binds $x to the entire list 'expr'
  - Useful for common subexpressions and aggregations

17

FOR vs LET

```
FOR $x IN document("bib.xml")/bib/book
RETURN <result> $x </result>
```

Returns:

```
<result> <book>...</book></result>
<result> <book>...</book></result>
<result> <book>...</book></result>
...
```

```
LET $x IN document("bib.xml")/bib/book
RETURN <result> $x </result>
```

Returns:

```
<result> <book>...</book>
         <book>...</book>
         <book>...</book>
         ...
</result>
```

18

FOR Example, …

- Find all book titles published after 1995

```
FOR $x IN document("bib.xml")/bib/book
WHERE $x/year > 1995
RETURN $x/title
```

Result:

```
<title> abc </title>
<title> def </title>
<title> ghi </title>
```

19

Nesting FORs

- For each author of a book published by Morgan Kaufman, list all books published:

```
FOR $a IN distinct(document("bib.xml")
                   /bib/book[publisher="Morgan Kaufmann"]/author)
RETURN <result>
          $a,
          FOR $t IN /bib/book[author=$a]/title
          RETURN $t
       </result>
```

distinct = a function that eliminates duplicates

20

Query Results

```
<result>
      <author>Jones</author>
      <title> Weaving the Web </title>
      <title> Databases: An Introduction </title>
</result>
<result>
      <author> Smith </author>
      <title>My Kitchen's Aflame! </title>
</result>
```

21

Let's add more

```
<big_publishers>
    FOR $p IN distinct(document("bib.xml")//publisher)
    LET $b := document("bib.xml")/book[publisher = $p]
    WHERE count($b) > 100
    RETURN $p
</big_publishers>
```

count = an (aggregate) function that returns the number of elts

What does this do?

22

Aggregates

- How can we find books with price greater than the average?
- Assume you have avg()

```
LET $a=avg(document("bib.xml")/bib/book/price)
FOR $b in document("bib.xml")/bib/book
WHERE $b/price > $a
RETURN $b
```

23

More FOR vs LET

- FOR binds to node variables
  - Allows iteration
- LET binds to collection varibles
  - Yields single value

24

Ordering

- Collections may be ordered or unordered
  - Ordered: `/bib/book/author`
  - Unordered: `distinct(/bib/book/author)`
- `LET $a = /bib/book ->` $a is a collection
- `$b/author` -> a collection (of authors)

```
RETURN <result> $b/author </result>
```

Returns:

```
<result> <author>...</author>
         <author>...</author>
         <author>...</author>
         ...
</result>
```

25

Collections

- What happens to collections in exprs?
  - $b/price ⇒ list of *n* prices
  - $b/price * 0.7 ⇒ ???
  - $b/price * $b/quantity ⇒ ?????????
- Valid only if the two sequences have at most one element; called *atomization*
- Careful when comparing items!
  - Value comparison eq will test atomic values
  - General comp. = tests sequences, atomic values,
- Compare:
  - $book1/author eq "Kennedy"
  - $book1/author = "Kennedy"
- What does = do on a sequence?

26

Sorting

```
<publisher_list>
  FOR $p IN distinct(document("bib.xml")//publisher)
  ORDERBY  $p
  RETURN <publisher> <name> $p/text() </name> ,
           FOR $b IN document("bib.xml")//book[publisher = $p]
           ORDERBY $b/price DESCENDING
           RETURN <book>
                    $b/title ,
                    $b/price
                  </book>
         </publisher>
</publisher_list>
```

27

Conditionals

```
FOR $h IN //holding
ORDERBY $h/title
RETURN <holding>
         $h/title,
         IF $h/@type = "Journal"
           THEN $h/editor
         ELSE $h/author
       </holding>
```

28

Existential Quantifiers

```
FOR $b IN //book
WHERE SOME $p IN $b//para SATISFIES
  contains($p, "sailing")
  AND contains($p, "windsurfing")
RETURN $b/title
```

29

Universal Quantifiers

```
FOR $b IN //book
WHERE EVERY $p IN $b//para SATISFIES
  contains($p, "sailing")
RETURN $b/title
```

30

Other XQuery Stuff

- Before and After
  - Dealing with order in the input
- Filter
  - Deletes some edges in the result tree
- Recursive functions

31

Group-By in XQuery

- No GROUPBY in XQuery
- A recent proposal
  - Ideas?

32

GROUPBY in XQuery?

```
FOR $b IN document("http://www.bn.com")/bib/book,
    $y IN $b/@year
WHERE $b/publisher="Morgan Kaufmann"
RETURN    GROUPBY $y
          WHERE count($b) > 10
          IN <year> $y </year>
```

← with GROUPBY

Equivalent SQL →

```
SELECT year
FROM Bib
WHERE Bib.publisher="Morgan Kaufmann"
GROUPBY year
HAVING count(*) > 10
```

33

GROUPBY in XQuery?

```
FOR $b IN document("http://www.bn.com")/bib/book,
    $a IN $b/author,
    $y IN $b/@year
RETURN  GROUPBY $a, $y
        IN <result> $a,
                   <year> $y </year>,
                   <total> count($b) </total>
      </result>
```

← with GROUPBY

Without GROUPBY →

```
FOR $a IN document("http://www.bn.com")/ bib/book/author,
    $y IN $a/../@year
LET $b = document("http://www.bn.com")/bib/book[author=$a,@year=$y]
RETURN <result> $a,
                <year> $y </year>,
                <total> count($b) </total>
       </result>
```

Correct if the GROUPBY is node-identity based
Not equivalent if the GROUPBY is value-based

34

GROUPBY in XQuery?

```
FOR $b IN document("http://www.bn.com")/bib/book,
    $a IN $b/author,
    $y IN $b/@year
RETURN  GROUPBY $a, $y
        IN <result> $a,
                   <year> $y </year>,
                   <total> count($b) </total>
      </result>
```

← with GROUPBY

Without →

```
FOR $a IN distinct(document("http://www.bn.com")/ bib/book/author)
    $y IN distinct(document("http://www.bn.com")/bib/book/@year)
LET $b =
document("http://www.bn.com")/bib/book[author=$a,@year=$y]
RETURN
        IF count($b) > 0 THEN
                <result> $a,
                        <year> $y </year>,
                                <total> count($b) </total>
                        </result>
```

35

GROUPBY in XQuery

```
FOR $b IN document("http://www.bn.com")/bib/book,
    $a IN $b/author,
    $y IN $b/@year
RETURN  GROUPBY $a, $y
        IN <result> $a,
                   <year> $y </year>,
                   <total> count($b) </total>
      </result>
```

← with GROUPBY

Without →

```
FOR $Tup IN distinct(FOR $b IN document("http://www.bn.com")/bib,
                $a IN $b/author,
                $y IN $b/@year
            RETURN <Tup> <a> $a </a> <y> $y </y> </Tup>),
    $a IN $Tup/a/node(),
    $y IN $Tup/y/node()
LET $b = document("http://www.bn.com")/bib/book[author=$a,@year=$y]
RETURN <result> $a,
                <year> $y </year>,
                <total> count($b) </total>
       </result>
```

36

GROUPBY in XQuery

```
FOR $b IN document("http://www.bn.com")/bib/book,
      $a IN $b/author,
      $y IN $b/@year,
      $t IN $b/title,
      $p IN $b/publisher
RETURN
   GROUPBY $p, $y
   IN <result> $p,
            <year> $y </year>,
            GROUPBY $a
             IN <authorEntry>
                        $a,
                        GROUPBY $t
                        IN $t
                 <authorEntry>
      </result>
```

← Nested GROUPBY's

37

Try this!

- http://sig.biostr.washington.edu/projects/wix/wix-demo.html

38

Part I

- The Server

39