CIS 330: Applied Database Systems

Lecture 25: XML Schema and XQuery

Johannes Gehrke
johannes@cs.cornell.edu
http://www.cs.cornell.edu/johannes

Some slides courtesy of Dan Suciu.

Lecture Overview

- Two topics today:
  - XML Schema
  - XQuery

XML Schema

- Schema: Defines class of XML documents
- Instance: XML document that conforms to the schema

- http://apps.gotdotnet.com/xmltools/xsdvalidator/

Running Example: Purchase Order

- Show po.xml
- Show po.xsd

- Elements:
  - schema
  - element
  - complexType
  - simpleType

XML Types

- Complex types:
  - Can contain other elements
  - Can have attributes
- Simple types:
  - No element content
  - No attributes

- Let's start with complex types

Complex Types: USAddress Type

```
<xsd:complexType name="USAddress" >
  <xsd:sequence>
    <xsd:element name="name"   type="xsd:string"/>
    <xsd:element name="street" type="xsd:string"/>
    <xsd:element name="city"   type="xsd:string"/>
    <xsd:element name="state"  type="xsd:string"/>
    <xsd:element name="zip"    type="xsd:decimal"/>
  </xsd:sequence>
  <xsd:attribute name="country" type="xsd:NMTOKEN" fixed="US"/>
 </xsd:complexType>
```

- Contains only simple types
- Note: Attributes must be simple types

Complex Types: PurchaseOrder Type

```
<xsd:complexType name="PurchaseOrderType">
 <xsd:sequence>
  <xsd:element name="shipTo" type="USAddress"/>
  <xsd:element name="billTo" type="USAddress"/>
  <xsd:element ref="comment" minOccurs="0"/>
  <xsd:element name="items"  type="Items"/>
 </xsd:sequence>
 <xsd:attribute name="orderDate" type="xsd:date"/>
</xsd:complexType>
```

- Contains both simple and complex types
- Ref element: refers to an existing element (must be a global element, not part of a complex type)

Occurrence Constraints

On Elements:

- <xsd:element ref="comment" minOccurs="0"/>
- Constraints:
  - minOccurs, maxOccurs

On Attributes:

- <xsd:attribute name="partNum" type="SKU" use="required"/>
- Use attribute values:
  - Required, optional, prohibited

Default and Fixed Values

- Exist for both elements and attributes

Default values:

- Default values for attributes:
  - The attribute has the default value
- Default values for elements:
  - An empty element has the default value

Fixed values:

- If value exists, it must be the default value
- Usage of both fixed and default is a mistake

Table 1. Occurrence Constraints for Elements and Attributes

Elements (minOccurs, maxOccurs) fixed, default	Attributes use, fixed, default	Notes
(1, 1) -, -	required, -, -	element/attribute must appear once, it may have any value
(1, 1) 37, -	required, 37, -	element/attribute must appear once, its value must be 37
(2, unbounded) 37, -	n/a	element must appear twice or more, its value must be 37; in general, minOccurs and maxOccurs values may be positive integers, and maxOccurs value may also be "unbounded"
(0, 1) -, -	optional, -, -	element/attribute may appear once, it may have any value
(0, 1) 37, -	optional, 37, -	element/attribute may appear once, if it does appear its value must be 37, if it does not appear its value is 37
(0, 1) -, 37	optional, -, 37	element/attribute may appear once; if it does not appear its value is 37, otherwise its value is that given
(0, 2) -, 37	n/a	element may appear once, twice, or not at all; if the element does not appear it is not provided; if it does appear and it is empty, its value is 37; otherwise its value is that given; in general, minOccurs and maxOccurs values may be positive integers, and maxOccurs value may also be "unbounded"
(0, 0) -, -	prohibited, -, -	element/attribute must not appear

Note that neither minOccurs, maxOccurs, nor use may appear in the declarations of global elements and attributes.

Global Elements and Attributes

```
<xsd:element name="comment" type="xsd:string"/>
...
<xsd:element ref="comment"   minOccurs="0"/>
```

Global elements and attributes:

- They are children of the schema element.
- Can be referred to using the ref attribute.
- Cannot contain references themselves.
- Cannot contain
  - minOccurs, maxOccurs, use

Naming Conflicts

- Two elements within different types can have the same name

Simple Types: A Subset

Simple Type	Examples (delimited by commas)
string	Confirm this is electric
normalizedString	Confirm this is electric
token	Confirm this is electric
byte	-1, 126
unsignedByte	0, 126
base64Binary	GpM7
hexBinary	0FB7
integer	-126789, -1, 0, 1, 126789
positiveInteger	1, 126789
negativeInteger	-126789, -1
nonNegativeInteger	0, 1, 126789
nonPositiveInteger	-126789, -1, 0
int	-1, 126789675
unsignedInt	0, 1267896754
long	-1, 12678967543233
unsignedLong	0, 12678967543233
short	-1, 12678
unsignedShort	0, 12678
decimal	-1.23, 0, 123.4, 1000.00

Creation of New Simple Types

- Derive from existing simple types
- Examples:

```
<xsd:simpleType name="myInteger">
  <xsd:restriction base="xsd:integer">
    <xsd:minInclusive value="10000"/>
    <xsd:maxInclusive value="99999"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="SKU">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="\d{3}-[A-Z]{2}"/>
  </xsd:restriction>
</xsd:simpleType>
```

(3 digits, hyphen, two uppercase letters)

Creation of New Simple Types (Contd.)

- Enumerate all possible values
- Example:

```
<xsd:simpleType name="USState">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="AK"/>
    <xsd:enumeration value="AL"/>
    <xsd:enumeration value="AR"/>
    <!-- and so on ... -->
  </xsd:restriction>
</xsd:simpleType>
```

Simple Types (Contd.)

- Types can be:
  - Atomic (so far)
  - List types (we already know NMTOKENS, IDREFS)

```
<xsd:simpleType name="listOfMyIntType">
  <xsd:list itemType="myInteger"/>
</xsd:simpleType>
<listOfMyInt>20003 15037 95977
  95945</listOfMyInt>
```

  - List item is delimited by white space

List Types (Contd.)

```
<xsd:simpleType name="USStateList">
 <xsd:list itemType="USState"/>
</xsd:simpleType>

<xsd:simpleType name="SixUSStates">
 <xsd:restriction base="USStateList">
  <xsd:length value="6"/>
 </xsd:restriction>
</xsd:simpleType>

<sixStates>PA NY CA NY LA AK</sixStates>
```

Simple Types: Union Types

```
<xsd:simpleType name="zipUnion">
  <xsd:union memberTypes="USState
   listOfMyIntType"/>
</xsd:simpleType>
```

Valid instances:

- <zips>CA</zips>
- <zips>95630 95977 95945</zips>
- <zips>AK</zips>

Lecture Overview

- Two topics today:
  - XML Schema
  - XQuery

XQuery

- http://www.w3.org/XML/Query
- Design influences:
  - Compatibility with XML Schema, XSLT, XPath
  - Superset of XPath

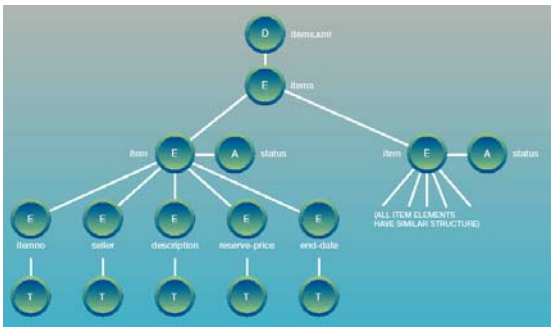
XQuery Data Model

- Sequence: Ordered collection of items
- Item: Node or atomic value
- Atomic value: Built-in data type from XML Schema
- Nodes: 7 types
  - Element, attribute, text, document, comment, processing instructions, and namespace
  - Can have recursive structure

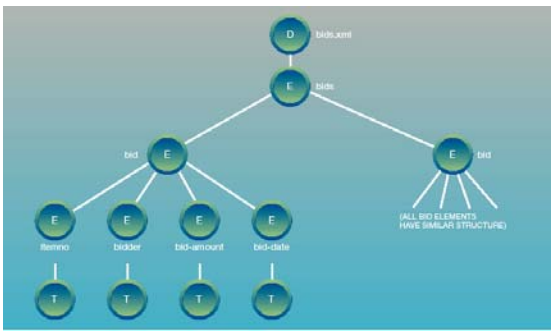
XQuery Data Model (Contd.)

- Element and attribute nodes:
  - Have typed values and/or names
  - Typed value: sequence of >= atomic values
- Nodes have identity
- Within a document, there is a total order, the document order (inorder traversal): node appears before its children

XQuery Data Model (Contd.)



XQuery Data Model (Contd.)



XQuery: Expressions

- XQuery is case sensitive, all keywords are lowercase
- Functional language
  - Expressions return values, no side effects
  - Wherever an expression occurs, any kind of expression is permissible
  - Value of an expression is heterogeneous sequence of nodes and atomic values.

XQuery: Expressions (Contd.)

- Literals

`47`	is a literal of type `integer`
`4.7`	is a literal of type `decimal` because it contains a decimal point
`4.7E3`	is a literal of type `double` because it contains an exponent
`"47"`	is a literal of type `string` (single quotes are allowed inside double-quoted strings)
`'47'`	is a literal of type `string` (double quotes are allowed inside single-quoted strings)

- Constructors
  - date("2002 05 31")
- Arithmetic expressions

XQuery: Expressions (Contd.)

- Sequences

`1, 2, 3`	is a sequence of three values
`(1, 2, 3)`	is identical to `1, 2, 3`
`((1, 2), (), 3)`	is identical to `1, 2, 3`
`1 to 3`	is identical to `1, 2, 3`

- Variables through LET expressions (more later)
- Function calls
  - substring("CS330",1,2)

XQuery: Expressions (Contd.)

- Path Expressions
- Examples:
  - *(Q1) List the descriptions of all items offered for sale by Smith.*
  - *(Q2) List all description elements found in the document items.xml.*
  - *(Q3) Find the status attribute of the item that is the parent of a some description.*

XQuery: Path Expressions

- Path Expressions
- Examples:
  - *(Q1) List the descriptions of all items offered for sale by Smith.*
    //item[seller="Smith"]/description*
  - *(Q2) List all description elements found in the document items.xml.*
    //description
  - *(Q3) Find the status attribute of the item that is the parent of a some description.*
    //description/../@status

XQuery: Predicates

- A predicate is an expression in square brackets that filters a sequence of values
  - item[seller = "Smith"]
  - item[reserve-price > 1000]
  - item[4]
  - item[reserve-price]
- Comparison operators
  - eq, ne, lt, le, gt, ge
  - =, !=, >, <=, <, <=
  - item[reserve-price gt 1000]

XQuery: Predicates (Contd.)

- Node comparison: is and isnot
- Order comparison: <<
- Logical operators: and, or, not
  - item[not(reserve price)]
  - item[seller eq "Smith" and reserve price]

XQuery: Element Constructors

- First choice: Just write XML
  - Use variables that are bound in an enclosing expression:

```
<highbid status = "{$s}">
   <itemno> {$i} </itemno>
   <bid-amount>
      {max($bids[itemno = $i]/bid-amount)}
   </bid-amount>
</highbid>
```

```
<highbid>
   {
    $b/@status,
    $b/itemno,
    $b/bid-amount
   }
</highbid>
```

XQuery: Element Constructors (Contd.)

- Keyword element expr1 expr2
  - expr1: computes the name of the element
  - expr2: computes the content of the element
  - Example:
    - element {name($e)}
      {$e/@*, data($e)*2}
- Similarly attribute constructors
  - Example:
    - attribute {if $p/sex="M" then "father" else "mother"} {$p/name}

XQuery: Iteration

- Examples:
  - for $m in (2,3), $n in (5,10)
    return <fact> {$m} times {$n} is
    {$m * $n}
    </fact>
- let versus for
  - let binds each variable to the associated sequence
  - for iterates each variable over the associated sequence

FLWR ("Flower") Expressions

FOR ... LET... FOR... LET...
WHERE...
RETURN...

XQuery

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>

XQuery

For each author of a book by Morgan Kaufmann, list all books she 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

XQuery

Result:

```
<result>
    <author>Jones</author>
    <title> abc </title>
    <title> def </title>
</result>
<result>
     <author> Smith </author>
     <title> ghi </title>
</result>
```

XQuery

- FOR $x in expr -- binds $x to each element in the list expr
- LET $x = expr -- binds $x to the entire list expr
  - Useful for common subexpressions and for aggregations

XQuery

```
<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 = a (aggregate) function that returns the number of elms

XQuery

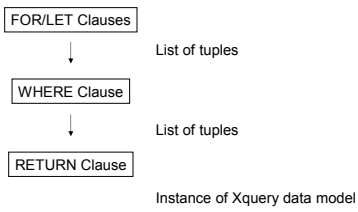
Find books whose price is larger than average:

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

XQuery

Summary:

- FOR-LET-WHERE-RETURN = FLWR



FOR v.s. LET

FOR
- Binds *node variables* → iteration

LET
- Binds *collection variables* → one value

FOR v.s. 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 := document("bib.xml")/bib/book
RETURN <result> $x </result>
```

Returns:

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

Collections in XQuery

- Ordered and unordered collections
  - /bib/book/author = an ordered collection
  - Distinct(/bib/book/author) = an unordered collection
- LET $a = /bib/book → $a is a collection
- $b/author → a collection (several authors...)

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

Returns:

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

Sorting in XQuery

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

Sorting in XQuery

- Sorting arguments: refer to the name space of the RETURN clause, not the FOR clause
- To sort on an element you don't want to display, first return it, then remove it with an additional query.

If-Then-Else

```
FOR $h IN //holding

RETURN <holding>

            $h/title,

            IF $h/@type = "Journal"

                  THEN $h/editor

            ELSE $h/author

      </holding> SORTBY (title)
```

Existential Quantifiers

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

Universal Quantifiers

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

Collections in XQuery

What about collections in expressions ?

- $b/@price → list of n prices
- $b/@price * 0.7 → list of n numbers
- $b/@price * $b/@quantity → list of n x m numbers ??
- $b/@price * ($b/@quant1 + $b/@quant2) ≠ $b/@price * $b/@quant1 + $b/@price * $b/@quant2 !!