JACOBS UNIVERSITY

XML, XPath, XQuery

Ramakrishnan & Gehrke, Chapter 24 / 27

Instructors: Peter Baumann

email: p.baumann@jacobs-university.de

tel: -3178

office: room 88, Research 1

XML

- XML = eXtensible Markup Language
  - flexible mechanism for defining domain-specific data exchange formats
  - ASCII

```
<molecule>
        <weight>234.5</weight>
        <spectra>…</spectra>
        <figures>…</figures>
</molecule>
```

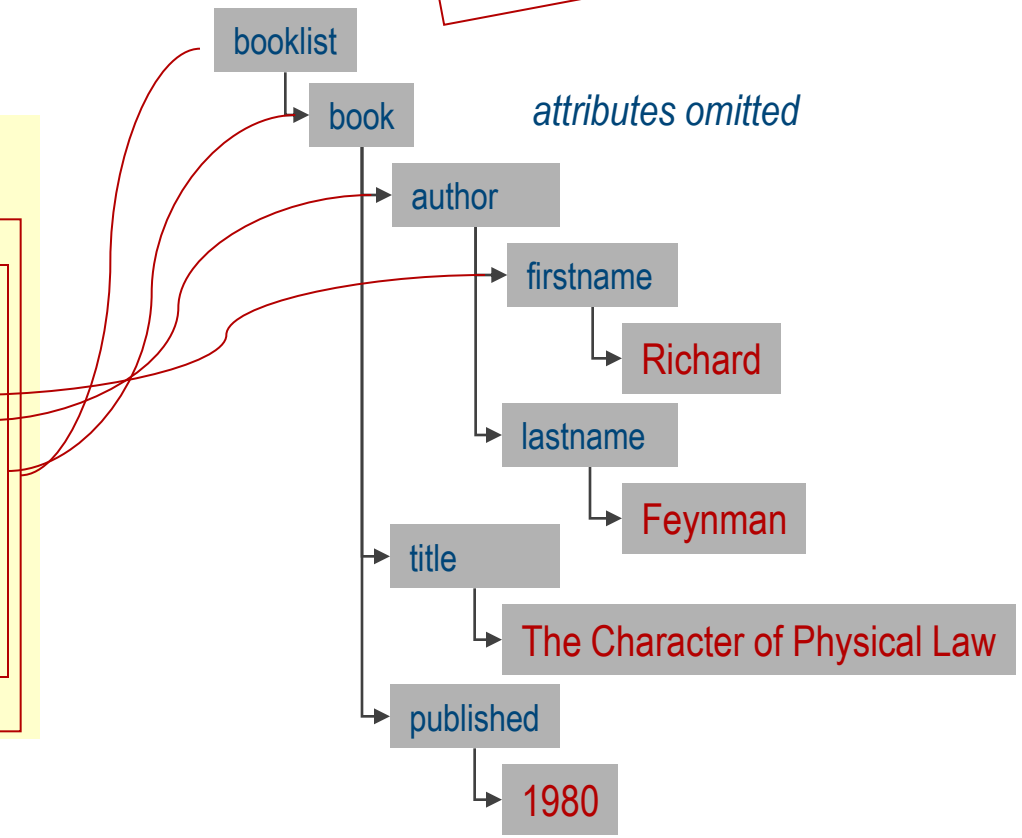
- "Extensible":
  meta language for defining new markup languages
  - Each language defines aka "document type", expressed in XML Schema
  - Ex: XHTML = HTML in XML; SensorML; MathML; MusicML; …
  - Automatic validity checking against schema

XML Document Tree

- Tree nodes = [element | attribute | text] infoset items

www.w3schools.com

```
<?xml version="1.0" encoding="UTF-8">
<!DOCTYPE booklist SYSTEM "library.std">
<booklist>
  <book genre="Science" format="Hardcover">
    <author>
      <firstname>Richard</firstname>
      <lastname>Feynman</lastname>
    </author>
    <title>The Character of Physical Law</title>
    <published>1980</published>
  </book>
</booklist>
```



XML Schema

- W3C Recommendation (ie: std), 2012
  - Schema for XML document instances, expressed in XML
  - extensible, built-in data type support, modular through namespaces
- Ex:

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.w3schools.com"
xmlns="http://www.w3schools.com"
elementFormDefault="qualified">

<xs:element name="note">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="to" type="xs:string"/>
      <xs:element name="from" type="xs:string"/>
      <xs:element name="heading" type="xs:string"/>
      <xs:element name="body" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```

„complex“ = contains other elements

„simple“ = no sub-elements

```
<?xml version="1.0" encoding="UTF-8"?>
<note xmlns="http://w3schools.com"
      xmls:xsi="http://w3.org/2001/XMLSchema-instance"
      xsi:schemalocation="http://www.w3schools.com note.xsd">
  <to>sample recicipient</to>
  <from>sample sender</from>
  <heading>as per phone call</heading>
  <body>Dear X, confirming our phone agreement. Yours, Y</body>
</note>
```

XML: A QL Perspective

- From a data modelling viewpoint, what does XML offer?
- Entities (ER!)
  - Recursively nested
- Attributes
  - Single-valued, atomic
- Relationships? Not really.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<catalog>
  <cd country="USA">
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <price>10.90</price>
  </cd>
  <cd country="UK">
    <title>Hide your heart</title>
    <artist>Bonnie Tyler</artist>
    <price>9.90</price>
  </cd>
  <cd country="USA">
    <title>Greatest Hits</title>
    <artist>Dolly Parton</artist>
    <price>9.90</price>
  </cd>
</catalog>
```

Pattern Expressions

- Walk document, return (sub)tree
  - „CD prices in catalog“:
    /catalog/cd/price
  - „all titles, artists“:
    /catalog/cd/title | /catalog/cd/artist
  - „all titles and artists“:
    //title | //artist
  - *„all CDs in catalog with price 10.90“*:
    /catalog/cd[price=10.90]
  - „all CD countries“:
    //cd/@country

```
<price>10.90</price>
<price>9.90</price>
<price>9.90</price>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<catalog>
  <cd country="USA">
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <price>10.90</price>
  </cd>
  <cd country="UK">
    <title>Hide your heart</title>
    <artist>Bonnie Tyler</artist>
    <price>9.90</price>
  </cd>
  <cd country="USA">
    <title>Greatest Hits</title>
    <artist>Dolly Parton</artist>
    <price>9.90</price>
  </cd>
</catalog>
```

XQuery

- XQuery – retrieving information from XML data
  - XQuery is to XML what SQL is to tables
  - XPath: extract from DOM tree; XQuery: derive new structure
  - Stored in files or in database
- FOR-LET-WHERE-ORDERBY-RETURN = FLWOR ("*flower*")
- Ex: "*all book titles published after 1995*"

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

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

Some More Capabilities

- aggregate functions: count, avg, …

```
<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>
```

```
<big_publishers>
 <publisher>Morgan Kaufmann</publisher>
 <publisher>Wiley</publisher>
</ big_publishers>
```

- Quantifiers: some, every, ...

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