Introduction to Semistructured Data and XML

Adapted from
Ramakrishnan and Gehrke

The Web a few years ago …

- HTML documents
  - often generated by applications
  - consumed by humans only
  - easy access: across platforms, across organizations
- No application interoperability:
  - HTML not understood by applications
    - screen scraping brittle
  - Database technology: client-server
    - still vendor specific

New Universal Data Exchange Format: XML

A recommendation from the W3C

- XML = data
- XML generated by applications
- XML consumed by applications
- Easy access: across platforms, organizations

Paradigm Shift on the Web

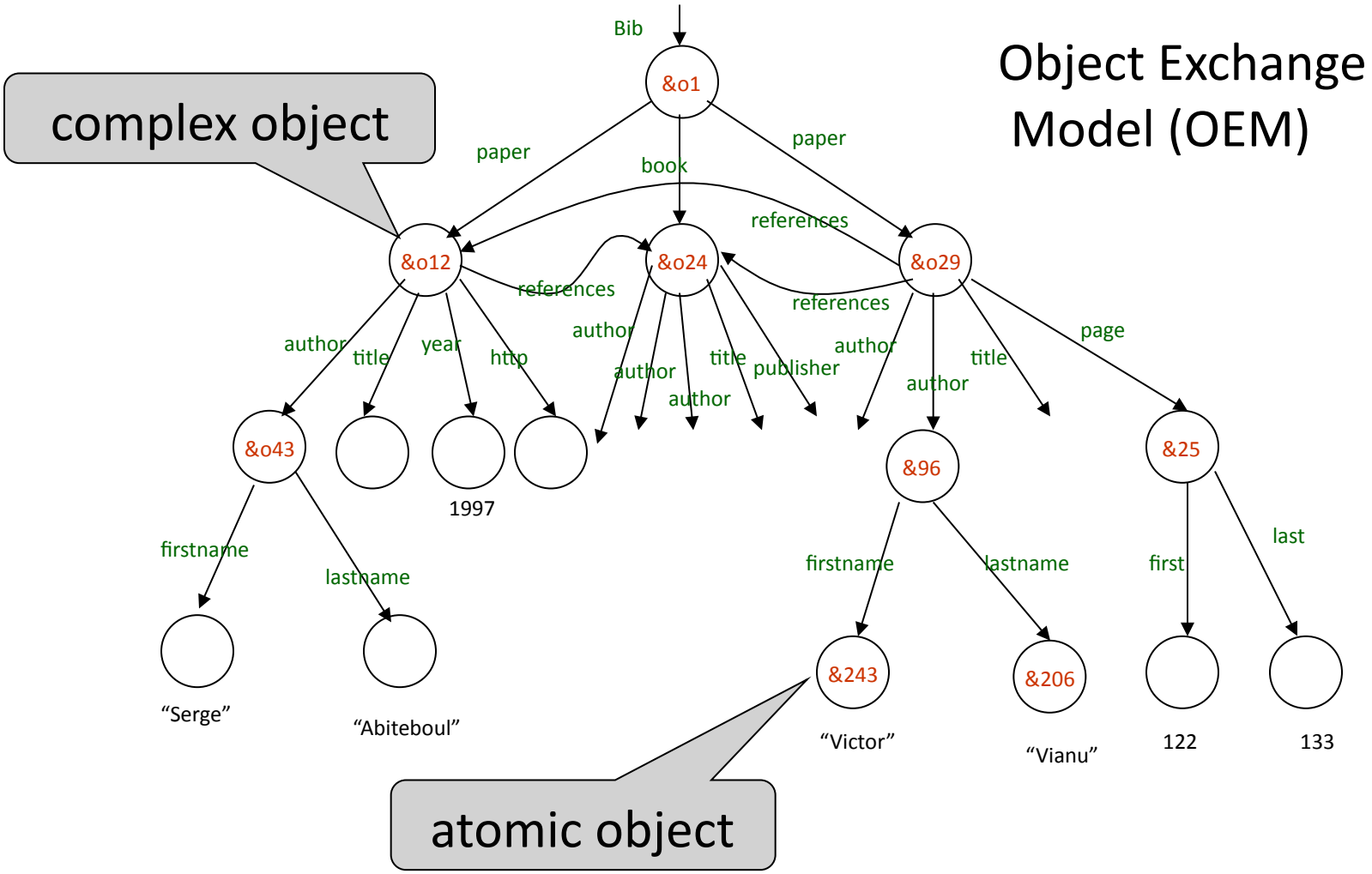
- From documents (HTML) to data (XML)
- From information retrieval to data management
- For databases, also a shift:
  - from relational model to semistructured data
  - from data processing to data/query translation
  - from storage to transport

Semistructured Data

Origins:

- Integration of heterogeneous sources
- Data sources with non-rigid structure
  - Biological data
  - *Web data*

The Semistructured Data Model



Syntax for Semistructured Data

```
Bib: &o1 { paper: &o12 { … },
           book:  &o24 { … },
           paper: &o29
                { author: &o52 "Abiteboul",
                  author: &o96 { firstname: &243 "Victor",
                                 lastname: &o206 "Vianu"},
                  title: &o93 "Regular path queries with constraints",
                  references: &o12,
                  references: &o24,
                  pages: &o25 { first: &o64 122, last: &o92 133}
                }
          }
```

Observe: Nested tuples, set-values, oids!

Syntax for Semistructured Data

May omit oids:

```
{ paper: { author: "Abiteboul",
           author: { firstname: "Victor",
                     lastname: "Vianu"},
           title: "Regular path queries …",
           page: { first: 122, last: 133 }
         }
}
```

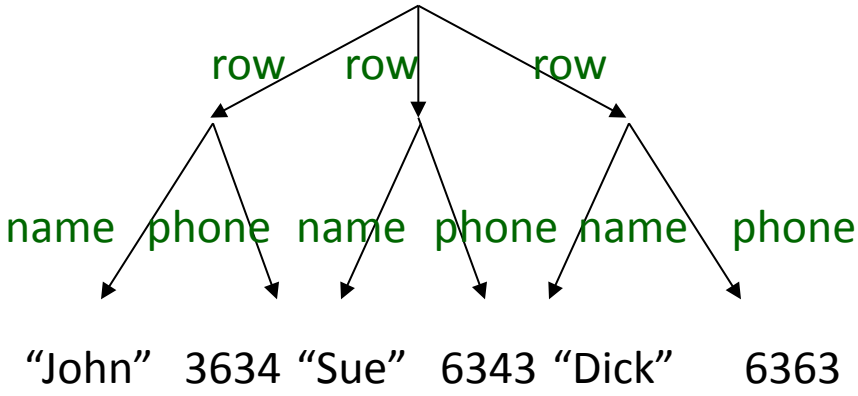
Characteristics of Semistructured Data

- Missing or additional attributes
- Multiple attributes
- Different types in different objects
- Heterogeneous collections

Self-describing, irregular data, no a priori structure

Comparison with Relational Data

name	phone
John	3634
Sue	6343
Dick	6363



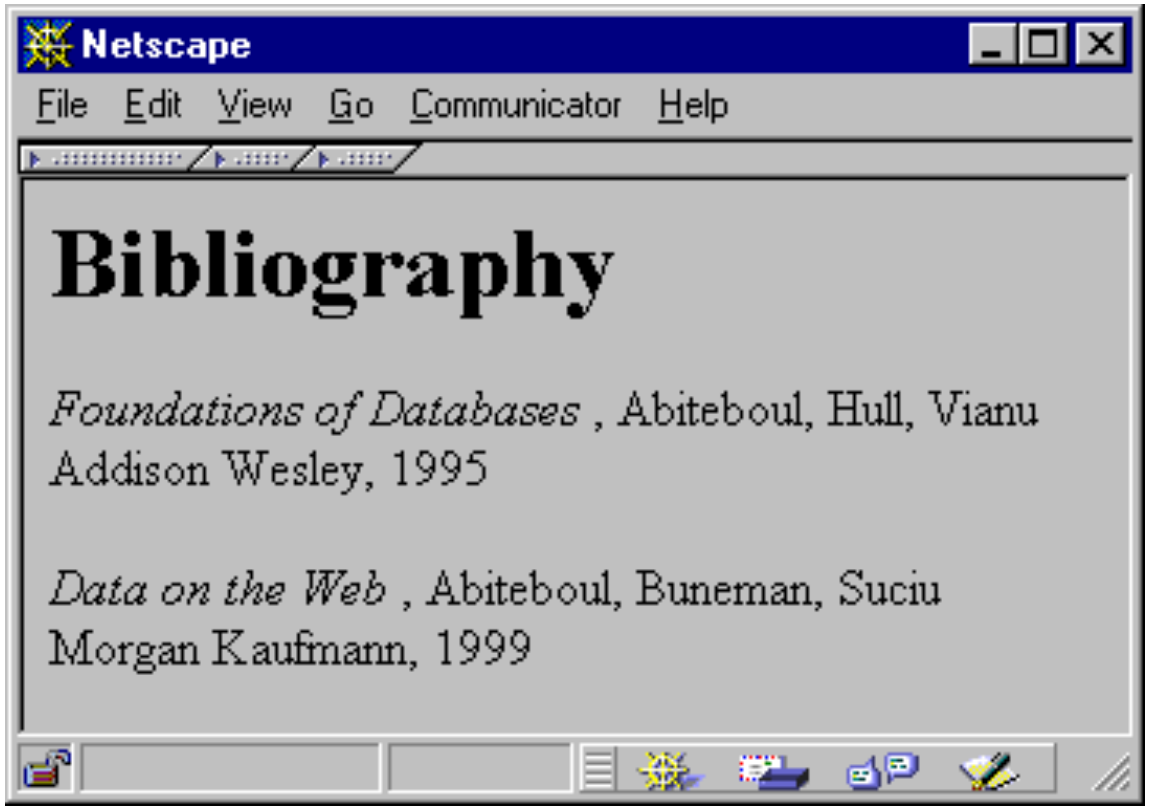
{ row: { name: "John", phone: 3634 },
  row: { name: "Sue", phone: 6343 },
  row: { name: "Dick", phone: 6363 }
}

XML

- A W3C standard to complement HTML
- Origins: Structured text SGML
  - Large-scale electronic publishing
  - Data exchange on the web
- Motivation:
  - HTML describes presentation
  - XML describes content
- http://www.w3.org/TR/2000/REC-xml-20001006 (version 2, 10/2000)

$$\text{HTML4.0} \in \text{XML} \subset \text{SGML}$$

From HTML to XML



HTML describes the presentation

HTML

```
<h1> Bibliography </h1>
<p> <i> Foundations of Databases </i>
        Abiteboul, Hull, Vianu
        <br> Addison Wesley, 1995 </br> </p>
<p> <i> Data on the Web </i>
        Abiteboul, Buneman, Suciu
        <br> Morgan Kaufmann, 1999 </br></p>
```

XML

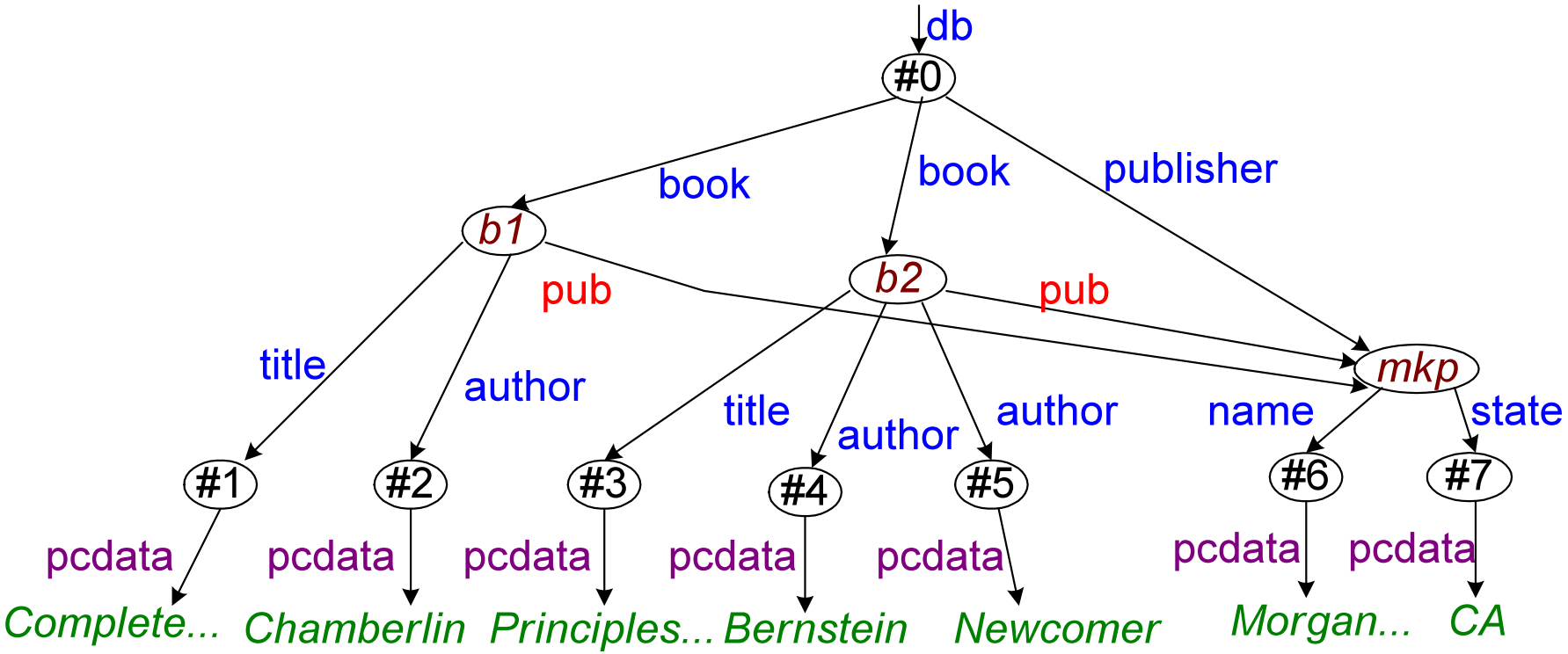
```
<bibliography>
    <book>   <title> Foundations… </title>
             <author> Abiteboul </author>
             <author> Hull </author>
             <author> Vianu </author>
             <publisher> Addison Wesley </publisher>
             <year> 1995 </year>
    </book>
    …
</bibliography>
```

XML describes the content

XML Terminology

- Tags: book, title, author, …
  - start tag: <book>, end tag: </book>
- Elements: <book>…</book>,<author>…</author>
  - elements can be nested
  - empty element: <red></red> (Can be abbrv. <red/>)
- XML document: Has a single root element
- Well-formed XML document: Has matching tags
- Valid XML document: conforms to a schema

XML Data Model (Graph)



More XML: Attributes

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

Attributes are alternative ways to represent data

XML Schema

- Schema definition for XML data
- In XML format
- Element names and types associated locally
- Includes primitive data types (integers, strings, dates, etc.)
- Supports value-based constraints (integers > 100)
- User-definable structured types
- Inheritance (extension or restriction)
- Foreign keys
- Element-type reference constraints

Sample XML Schema

```
<schema version="1.0" xmlns="http://www.w3.org/2001/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>
```

Important XML Standards

- XSL/XSLT: presentation and transformation standards
- RDF: resource description framework (meta-info such as ratings, categorizations, etc.)
- Xpath/Xpointer/Xlink: standard for linking to documents and elements within
- Namespaces: for resolving name clashes
- DOM: Document Object Model for manipulating XML documents
- SAX: Simple API for XML parsing
- XQuery: query language

XML vs. Semistructured Data

- Both described best by a graph
- Both are schema-less, self-describing
- XML is ordered, ssd is not
- XML can mix text and elements:

```
<talk> Making Java easier to type and easier to type
        <speaker> Phil Wadler </speaker>
</talk>
```

- XML has lots of other stuff: attributes, entities, processing instructions, comments

Path Expressions

Examples:

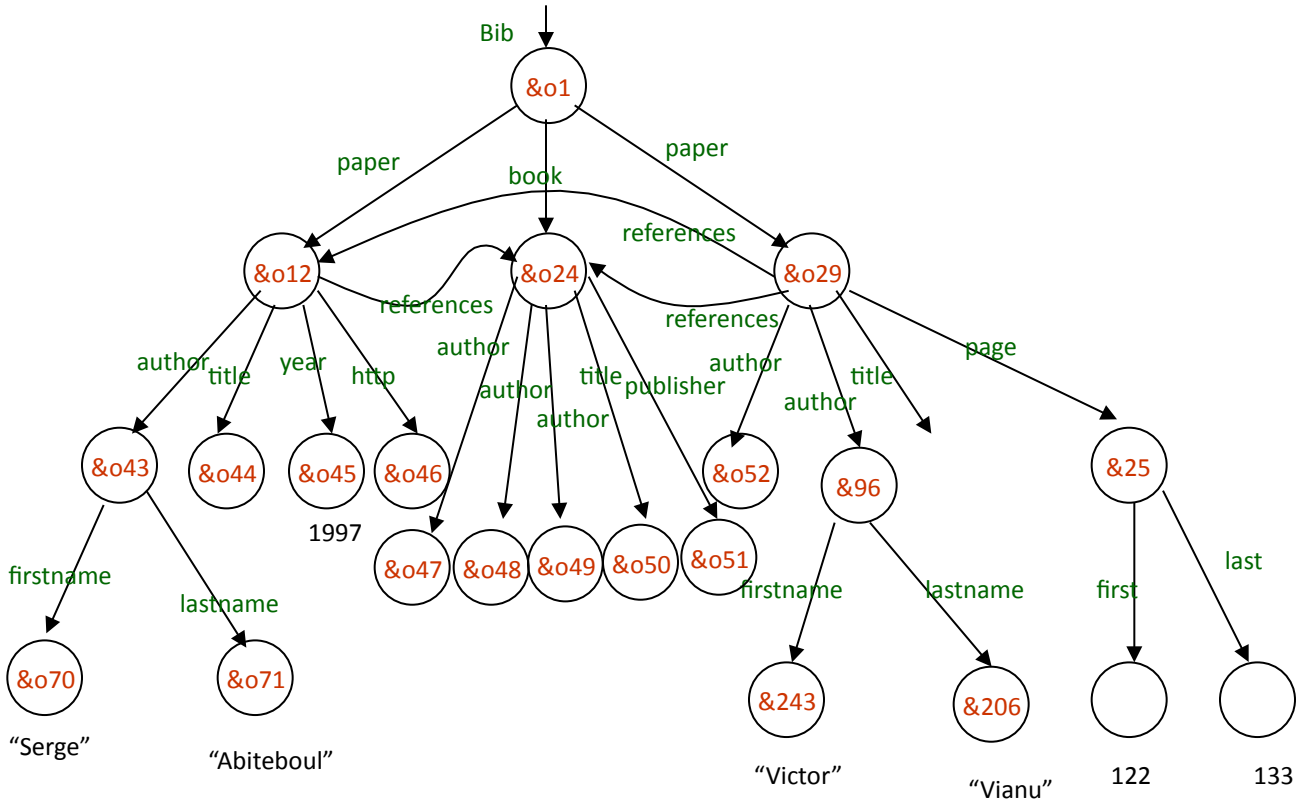
- Bib.paper
- Bib.book.publisher
- Bib.paper.author.lastname

Given an OEM instance, the *value* of a path expression *p* is a set of objects

Path Expressions

Examples:

DB =



Bib.paper={&o12,&o29}

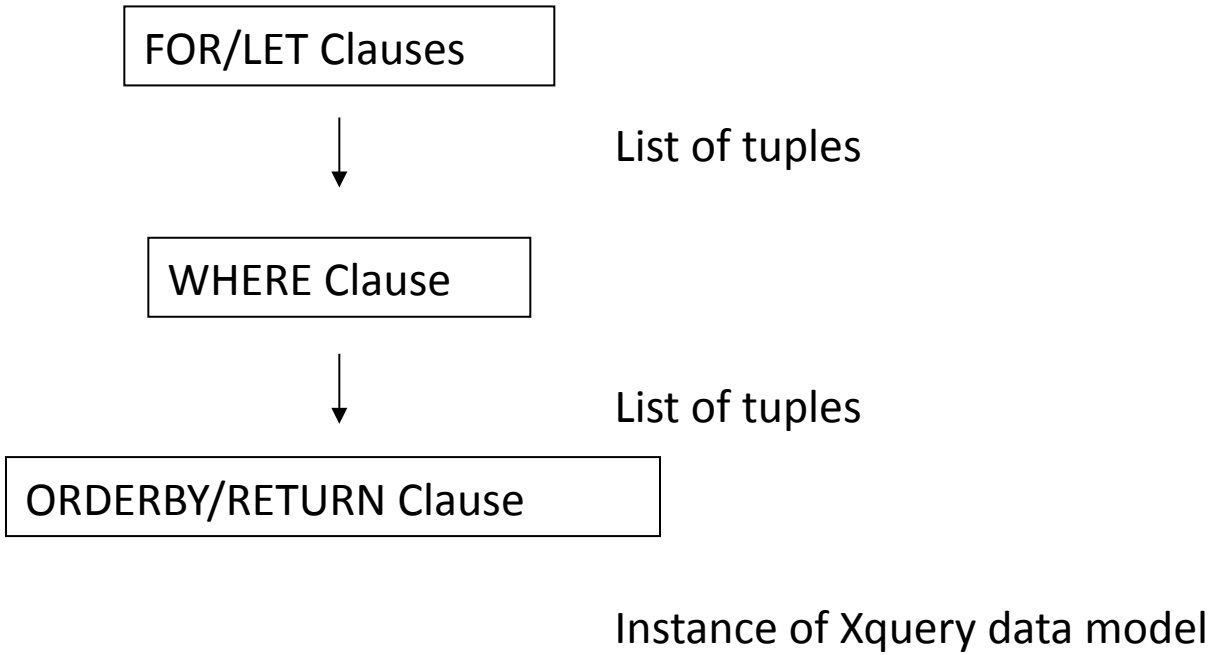
Bib.book.publisher={&o51}

Bib.paper.author.lastname={&o71,&206}

XQuery

Summary:

- FOR-LET-WHERE-ORDERBY-RETURN = FLWOR



XQuery

- 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 for aggregations

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

Returns:

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

Path Expressions

- Abbreviated Syntax
  - /bib/paper[2]/author[1]
  - /bib//author
  - paper[author/lastname=“Vianu"]
  - /bib/(paper|book)/title
- Unabbreviated Syntax
  - child::bib/descendant::author
  - child::bib/descendant-or-self::*/child::author
  - parent, self, descendant-or-self, attribute

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

```
<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
```

FOR v.s. LET

FOR

- Binds *node variables* → iteration

LET

- Binds *collection variables* → one value

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>
```

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 ??
  - Valid only if the two sequences have at most one element
  - Atomization
- $book1/author eq "Kennedy" - Value Comparison
- $book1/author = "Kennedy" - General Comparison

Sorting in XQuery

```
<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>
```

If-Then-Else

```
FOR $h IN //holding
ORDERBY $h/title
RETURN <holding>

                $h/title,

                IF $h/@type = "Journal"

                        THEN $h/editor

                ELSE $h/author

        </holding>
```

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
```

Some Other Stuff in XQuery

- If-then-else
- Universal and existential quantifiers
- Sorting
- Before and After
  - for dealing with order in the input
- Filter
  - deletes some edges in the result tree
- Recursive functions