Introduction to Semistructured Data and XML

Chapter 27, Part E
Based on slides by Dan Suciu
University of Washington

Management of XML and Semistructured Data

Based upon slides by Dan Suciu

Path Expressions

Examples:

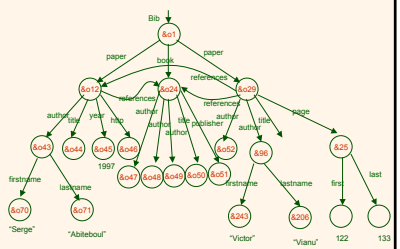
- Bib.paper
- Bib.book.publisher
- Bib.paper.author.lastname

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

Path Expressions

Examples:

DB =

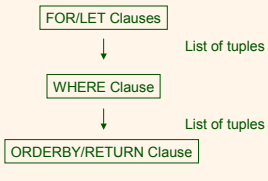


Bib.paper={&o12,&o29}
Bib.book.publisher={&o51}
Bib.paper.author.lastname={&o71,&206}

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
```


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

Group-By in Xquery ??

- No GROUPBY currently in XQuery
- A recent proposal (next)
  - What do YOU think ?

Group-By 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
```


Group-By 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

Group-By 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 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>
```


Group-By 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 $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>
```


Group-By 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