

# CS/INFO 330

## XQuery

October 23, 2005: Guest lecture by Al Demers  
October 26: Continuation by JG

Slides courtesy of Dan Suciu, University of Washington

1

---

---

---

---

---

---

---

---

## Announcements

- If you do not have your final exam:
  - Johannes will bring it on Wednesday to class
- Programming assignment is due on Wednesday at 11pm
- Design document
  - Due date: Wednesday, November 2 in class
  - Instructions are on the web

2

---

---

---

---

---

---

---

---

## Recall from Last Lecture: XPath

bib	matches a bib element
*	matches any element
/	matches the root element
/bib	matches a bib element under root
bib/paper	matches a paper in bib
bib//paper	matches a paper in bib, at any depth
//paper	matches a paper at any depth
paper book	matches a paper or a book
@price	matches a price attribute
bib/book/@price	matches price attribute in book, in bib
bib/book[@price<"55"]/author/lastname	matches...

3

---

---

---

---

---

---

---

---

## XQuery

- Based on Quilt, which is based on XML-QL
- Uses XPath to express more complex queries

4

---

---

---

---

---

---

---

---

## FLWR (“Flower”) Expressions

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

5

---

---

---

---

---

---

---

---

## Sample Data for Queries (more or less)

```
<bib>
  <book> <publisher> Addison-Wesley </publisher>
    <author> Serge Abiteboul </author>
    <author> <first-name> Rick </first-name>
      <last-name> Hull </last-name>
    </author>
    <author> Victor Vianu </author>
    <title> Foundations of Databases </title>
    <year> 1995 </year>
  </book>
  <book price="55">
    <publisher> Freeman </publisher>
    <author> Jeffrey D. Ullman </author>
    <title> Principles of Database and Knowledge Base Systems </title>
    <year> 1998 </year>
  </book>
</bib>
```

6

---

---

---

---

---

---

---

---

## FOR-WHERE-RETURN

Find all book titles published after 1995:

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

Result:

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

7

---

---

---

---

---

---

---

---

## FOR-WHERE-RETURN

Equivalently (perhaps more geekish)

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

And even shorter:

```
document("bib.xml")/bib/book[year/text() > 1995] /title
```

8

---

---

---

---

---

---

---

---

## FOR-WHERE-RETURN

- Find all book titles and the year when they were published:

```
FOR $x IN document("bib.xml")/bib/book
RETURN <answer>
  <what>{ $x/title/text() } </what>
  <when>{ $x/year/text() } </when>
</answer>
```

We can construct whatever XML results we want !

9

---

---

---

---

---

---

---

---

## Answer

```
<answer>
  <what> How to cook a Turkey </what>
  <when> 2003 </when>
</answer>
<answer>
  <what> Cooking While Watching TV </what>
  <when> 2004 </when>
</answer>
<answer>
  <what> Turkeys on TV</what>
  <when> 2002 </when>
</answer>
. . . .
```

10

---

---

---

---

---

---

---

---

## FOR-WHERE-RETURN

- Notice the use of “{“ and “}”
- What is the result without them ?

```
FOR $x IN document("bib.xml")/bib/book
RETURN <answer>
  <title> $x/title/text() </title>
  <year> $x/year/text() </year>
</answer>
```

11

---

---

---

---

---

---

---

---

## XQuery: Nesting

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

```
FOR $b IN document("bib.xml")/bib,
  $a IN $b/book[publisher/text()='Morgan Kaufmann']/author
RETURN <result>
  { $a,
    FOR $t IN $b/book[author/text()=$a/text()]/title
    RETURN $t
  }
</result>
```

In the **RETURN** clause comma concatenates XML fragments <sup>12</sup>

---

---

---

---

---

---

---

---

## XQuery

Result:

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

13

---

---

---

---

---

---

---

---

## Aggregates

Find all books with more than 3 authors:

```
FOR $x IN document("bib.xml")/bib/book
WHERE count($x/author)>3
RETURN $x
```

**count** = a function that counts  
**avg** = computes the average  
**sum** = computes the sum  
**distinct-values** = eliminates duplicates

14

---

---

---

---

---

---

---

---

## Aggregates

Same thing:

```
FOR $x IN document("bib.xml")/bib/book[count(author)>3]
RETURN $x
```

15

---

---

---

---

---

---

---

---

## Aggregates

Print all authors who published more than 3 books

```
FOR $b IN document("bib.xml")/bib,  
  $a IN $b/book/author/text()  
WHERE count($b/book[author/text()=$a])>3  
RETURN <author> { $a } </author>
```

What's wrong ?

16

---

---

---

---

---

---

---

---

## Aggregates

Be aware of duplicates !

```
FOR $b IN document("bib.xml")/bib,  
  $a IN distinct-values($b/book/author/text())  
WHERE count($b/book[author/text()=$a])>3  
RETURN <author> { $a } </author>
```

17

---

---

---

---

---

---

---

---

## XQuery

Find books whose price is larger than average:

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

LET binds a variable to one value;  
FOR iterates a variable over a list of values  
We will come back to that

18

---

---

---

---

---

---

---

---

## FOR-WHERE-RETURN

- “Flatten” the authors, i.e. return a list of (author, title) pairs

```
FOR $b IN document("bib.xml")/bib/book,  
  $x IN $b/title/text(),  
  $y IN $b/author/text()  
RETURN <answer>  
  <title> { $x } </title>  
  <author> { $y } </author>  
</answer>
```

Answer:  
<answer>  
 <title> abc </title>  
 <author> efg </author>  
</answer>  
<answer>  
 <title> abc </title>  
 <author> hkj </author>  
</answer>

19

---

---

---

---

---

---

---

---

## FOR-WHERE-RETURN

- For each author, return all book titles he/she wrote

```
FOR $b IN document("bib.xml")/bib,  
  $x IN $b/book/author/text()  
RETURN  
<answer>  
  <author> { $x } </author>  
  { FOR $y IN $b/book[author/text()=$x]/title  
    RETURN $y }  
</answer>
```

Answer:  
<answer>  
 <author> efg </author>  
 <title> abc </title>  
 <title> klm </title>  
 . . . .  
</answer>

What about  
duplicate  
authors ?

20

---

---

---

---

---

---

---

---

## FOR-WHERE-RETURN

- Same, but eliminate duplicate authors:

```
FOR $b IN document("bib.xml")/bib  
LET $a := distinct-values($b/book/author/text())  
FOR $x IN $a  
RETURN  
<answer>  
  <author> $x </author>  
  { FOR $y IN $b/book[author/text()=$x]/title  
    RETURN $y }  
</answer>
```

21

---

---

---

---

---

---

---

---

## FOR-WHERE-RETURN

- Same thing:

```
FOR $b IN document("bib.xml")/bib,  
  $x IN distinct-values($b/book/author/text())  
RETURN  
  <answer>  
  <author> $x </author>  
  { FOR $y IN $b/book[author/text()=$x]/title  
    RETURN $y }  
</answer>
```

22

---

---

---

---

---

---

---

---

## SQL and XQuery Side-by-side

Product(pid, name, maker, price) Find all product names, prices,  
sort by price

```
SELECT x.name,  
       x.price  
FROM Product x  
ORDER BY x.price
```

SQL

```
FOR $x in document("db.xml")/db/Product/row  
ORDER BY $x/price/text()  
RETURN <answer>  
  { $x/name, $x/price }  
</answer>
```

XQuery

23

---

---

---

---

---

---

---

---

## Answers

Name	Price
abc	7
def	23
...	...

```
<answer>  
  <name> abc </name>  
  <price> 7 </price>  
</answer>  
<answer>  
  <name> def </name>  
  <price> 23 </price>  
</answer>  
  ....
```

Notice: this is NOT a  
well-formed document !  
(WHY ???)

24

---

---

---

---

---

---

---

---



## Producing a Well-Formed Answer

```
<myQuery>
{ FOR $x in document("db.xml")/db/Product/row
  ORDER BY $x/price/text()
  RETURN <answer>
    { $x/name, $x/price }
    </answer>
}
</myQuery>
```

25

---

---

---

---

---

---

---

---

## Xquery's Answer

```
<myQuery>
<answer>
  <name> abc </name>
  <price> 7 </price>
</answer>
<answer>
  <name> def </name>
  <price> 23 </price>
</answer>
. . . .
</myQuery>
```

Now it is well-formed !

26

---

---

---

---

---

---

---

---

## SQL and XQuery Side-by-side

Product(pid, name, maker, price)  
Company(cid, name, city, revenues) Find all products made in Seattle

```
SELECT x.name
FROM Product x, Company y
WHERE x.maker=y.cid
and y.city="Seattle"
```

SQL

```
FOR $r in document("db.xml")/db,
  $x in $r/Product/row,
  $y in $r/Company/row
WHERE
  $x/maker/text()=$y/cid/text()
  and $y/city/text() = "Seattle"
RETURN { $x/name }
```

XQuery

```
FOR $y in /db/Company/row[city/text()='Seattle'],
  $x in /db/Product/row[maker/text()=$y/cid/text()]
RETURN { $x/name }
```

Cool XQuery

---

---

---

---

---

---

---

---

```

<product>
  <row> <pid> 123 </pid>
    <name> abc </name>
    <maker> efg </maker>
  </row>
  <row> .... </row>
  ...
</product>
<product>
  ...
</product>
....

```

28

---

---

---

---

---

---

---

---

---

---

### SQL and XQuery Side-by-side

For each company with revenues < 1M count the products over \$100

```

SELECT y.name, count(*)
FROM Product x, Company y
WHERE x.price > 100 and x.maker=y.cid and y.revenue < 1000000
GROUP BY y.cid, y.name

```

```

FOR $r in document("db.xml")/db,
  $y in $r/Company/row[revenue/text()<1000000]
RETURN
  <proudCompany>
    <companyName> { $y/name/text() } </companyName>
    <numberOfExpensiveProducts>
      { count($r/Product/row[maker/text()=$y/cid/text()][price/text()>100]) }
    </numberOfExpensiveProducts>
  </proudCompany>

```

---

---

---

---

---

---

---

---

---

---

### SQL and XQuery Side-by-side

Find companies with at least 30 products, and their average price

```

SELECT y.name, avg(x.price)
FROM Product x, Company y
WHERE x.maker=y.cid
GROUP BY y.cid, y.name
HAVING count(*) > 30

```

```

FOR $r in document("db.xml")/db,
  $y in $r/Company/row
LET $p := $r/Product/row[maker/text()=$y/cid/text()]
WHERE count($p) > 30
RETURN
  <theCompany>
    <companyName> { $y/name/text() }
    </companyName>
    <avgPrice> avg($p/price/text()) </avgPrice>
  </theCompany>

```

An element

A collection

30

---

---

---

---

---

---

---

---

---

---

## FOR v.s. LET

### FOR

- Binds *node variables* → iteration

### LET

- Binds *collection variables* → one value

31

---

---

---

---

---

---

---

---

## FOR v.s. LET

```
FOR $x IN /bib/book  
RETURN <result> { $x } </result>
```

Returns:  
<result> <book>...</book></result>  
<result> <book>...</book></result>  
<result> <book>...</book></result>  
...

```
LET $x := /bib/book  
RETURN <result> { $x } </result>
```

Returns:  
<result> <book>...</book>  
<book>...</book>  
<book>...</book>  
...  
</result>

32

---

---

---

---

---

---

---

---

## Collections in XQuery

- Ordered and unordered collections
  - `/bib/book/author/text()` = an *ordered* collection: result is in *document order*
  - `distinct-values(/bib/book/author/text())` = an *unordered* collection: the output order is implementation dependent
- `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>

33

---

---

---

---

---

---

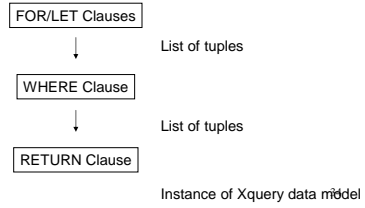
---

---

# XQuery

Summary:

- FOR-LET-WHERE-RETURN = FLWR



---

---

---

---

---

---

---

---