

Introduction to Database Systems CSE 444

Lecture 26: XQuery

CSE 444 – Autumn 2010

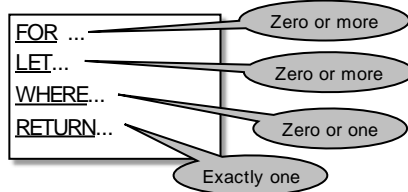
XQuery

- Standard for high-level querying of databases containing data in XML form
- Based on Quil, which is based on XML-QL
- Uses XPath to express more complex queries
- Readings
 - Section 12.2
 - [Nothing about XQuery in old Edition]

CSE 444 – Autumn 2010

2

FLWR (“Flower”) Expressions



CSE 444 – Autumn 2010

3

FOR-WHERE-RETURN

Find all book titles published after 1995:

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

Result:

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

CSE 444 – Autumn 2010

4

FOR-WHERE-RETURN

Equivalently (perhaps more geekish)

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

And even shorter:

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

CSE 444 – Autumn 2010

5

COERCION

The query:

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

Is rewritten by the system into:

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

CSE 444 – Autumn 2010

6

FOR-WHERE-RETURN

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

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

Result:

```
<answer> <title> abc </title> <year> 1995 </year> </answer>
<answer> <title> def </title> <year> 2002 </year> </answer>
<answer> <title> ghk </title> <year> 1980 </year> </answer>
```

CSE 444 – Autumn 2010

8

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>
```

CSE 444 – Autumn 2010

8

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>
```

```
<answer> <title> $x/title/text() </title> <year> $x/year/text() </year> </answer>
<answer> <title> $x/title/text() </title> <year> $x/year/text() </year> </answer>
<answer> <title> $x/title/text() </title> <year> $x/year/text() </year> </answer>
```

9

Nesting

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

10

Result

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

CSE 444 – Autumn 2010

11

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

CSE 444 – Autumn 2010

12

Aggregates

Same thing:

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

CSE 444 – Autumn 2010

13

Eliminating Duplicates

Print all authors:

```
FOR $a IN distinct-values($b/book/author/text())
RETURN <author> { $a } </author>
```

Note: distinct-values applies ONLY to values, NOT elements

CSE 444 – Autumn 2010

14

The LET Clause

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
```

CSE 444 – Autumn 2010

15

Flattening

- Compute a list of (author, title) pairs

Input:

```
<book>
  <title> Databases </title>
  <author> Widom </author>
  <author> Ullman </author>
</book>
```

Output:

```
<answer>
  <title> Databases </title>
  <author> Widom </author>
</answer>
<answer>
  <title> Databases </title>
  <author> Ullman </author>
</answer>
```

```
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>
```

16

Re-grouping

- For each author, return all titles of her/his books

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

Result:

```
<answer>
  <author> efg </author>
  <title> abc </title>
  <title> km </title>
  ...
</answer>
```

What about duplicate authors ?

CSE 444 – Autumn 2010

17

Re-grouping

- 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
  <author> $x </author>
  { FOR $y IN $b/book[author/text()=$x]/title
    RETURN $y }
</answer>
```

CSE 444 – Autumn 2010

18

Re-grouping

- 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>
```

CSE 444 – Autumn 2010

19

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

CSE 444 – Autumn 2010

20

XQuery's Answer

```
<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 ???)

CSE 444 – Autumn 2010

21

Producing a Well-Formed Answer

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

CSE 444 – Autumn 2010

22

XQuery's Answer

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

Now it is well-formed !

CSE 444 – Autumn 2010

23

SQL and XQuery Side-by-side

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

```
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

Cool XQuery

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

```

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

```

CSE 444 – Autumn 2010 25

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]) }
    </proudCompany>

```

SQL and XQuery Side-by-side

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

```

SELECT yname, avg(x.price)
FROM Product x, Company y
WHERE x.maker=y.cid
GROUP BY y.cid, yname
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