Introduction to Database Systems
CSE 444

Lecture 26: XQuery

XQuery

- Standard for high-level querying of databases containing data in XML form
- Based on Quilt, which is based on XML-QL
- Uses XPath to express more complex queries

Readings
- Section 12.2
- [Nothing about XQuery in old Edition]

FLWR (“Flower”) Expressions

FOR
LET
WHERE
RETURN

Zero or more
Zero or more
Zero or one
Exactly one

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>

FOR-WHERE-RETURN

Equivalent (perhaps more geekish)

RETURN $x$

And even shorter:


COERCION

The query:

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

Is rewritten by the system into:

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

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

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

Result:

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

FOR-WHERE-RETURN

• Notice the use of "{" and "}"
• What is the result without them?

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

In the RETURN clause comma concatenates XML fragments.

Result

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

FOR-WHERE-RETURN

• Notice the use of "{" and "}"
• What is the result without them?

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

Aggregates

Find all books with more than 3 authors:

```xml
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
Aggregates

Same thing:

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

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

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 $x$
```

Flattening

• Compute a list of (author, title) pairs

Input:
```
<book>
  <title> Databases </title>
  <author> Widom </author>
</book>
```

Output:
```
<answer>
  <title> Databases </title>
  <author> Widom </author>
</answer>
```

Re-grouping

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

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

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

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
```

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

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 ???)
```

Producing a Well-Formed Answer

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

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!

SQL and XQuery Side-by-side

Find all products made in Seattle

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

```
FOR $r$ in document("db.xml")/db
  $y$ in $r$/Company/row
WHERE $y$/city/text() = "Seattle"
  and $y$/city/text() = "Seattle"
RETURN ($y$/name)
```

```
FOR $y$ in db/Company/row[city/text()="Seattle"]
  $x$ in db/Product/row[maker/text()=$y$/cid/text()]
RETURN ($x$/name)
```
SQL and XQuery Side-by-side

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

```sql
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
```

```xquery
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>
```

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

```sql
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
```

```xquery
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>
```

A collection

An element