

Announcements (March 21)

- * Midterm has been graded
- Homework #3 will be assigned next Tuesday
- * Reading assignment due next Wednesday
 - XML processing in Lore (VLDB 1999) and Niagara (VLDB 2003)
- * Project milestone 2 due next Thursday

XQuery

- XPath + full-fledged SQL-like query language
- * XQuery expressions can be
 - XPath expressions
 - FLWR (❀) expressions
 - Quantified expressions
 - Aggregation, sorting, and more...
- An XQuery expression in general can return a new result XML document
 - Compare with an XPath expression, which always returns a sequence of nodes from the input document or atomic values (boolean, number, string, etc.)

A simple XQuery based on XPath

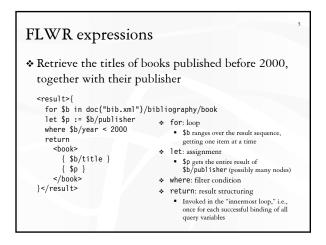
Find all books with price lower than \$50 <result>

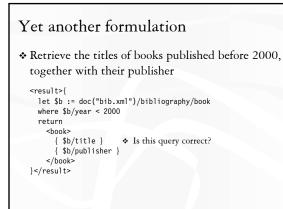
{

doc("bib.xml")/bibliography/book[@price<50]</pre>

</result>

- $\boldsymbol{\ast}$ Things outside { }'s are copied to output verbatim
- Things inside {}'s are evaluated and replaced by the results
 doc("bib.xml") specifies the document to query
 - The XPath expression returns a sequence of book elements
 - These elements (including all their descendents) are copied to output



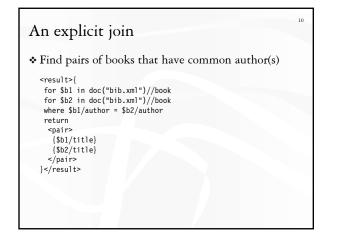


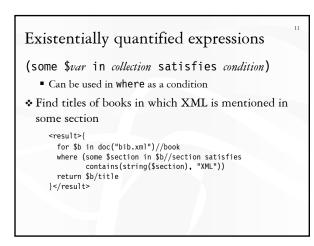
Subqueries in return

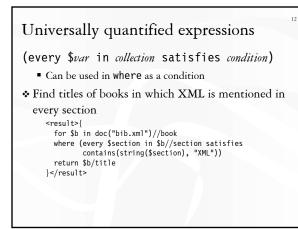
```
Extract book titles and their authors; make title an attribute and rename author to writer
```

```
<br/>
<br/>
<br/>
<br/>
for $b in doc("bib.xml")/bibliography/book
return
<br/>
<br/
```

 normalize-space(string) removes leading and trailing spaces from string, and replaces all internal sequences of white spaces with one white space







Aggregation

- * List each publisher and the average prices of all its books <result>{
 - for \$pub in distinct-values(doc("bib.xml")//publisher) let \$price := avg(doc("bib.xml")//book[publisher=\$pub]/@price)

13

14

15

- return
- cum
 sum
 <p </publisherpricing>
- }</result>
- distinct-values (collection) removes duplicates by value • If the collection consists of elements (with no explicitly declared types), they are first converted to strings representing their text contents
- avg(collection) computes the average of collection (assuming each item in collection can be converted to a numeric value)

Sorting (a brief history)

- * XPath always returns a sequence of nodes in original document order
- * for loop will respect the ordering in the sequence
- * August 2002
 - Introduce an operator sort by (sort-by-expression-list) to output results in a user-specified order
 - Example: list all books with price higher than \$100, in order by first author; for books with the same first author, order by title <result>{

doc("bib.xml")//book[@price>100] sort by (author[1], title) }</result>

Tricky semantics

* List titles of all books, sorted by their prices <result>{ (doc("bib.xml")//book sort by (@price))/title </result>

What is wrong?

Current version of sorting

As of March 2005

- * sort by has been ditched
- Add a new order by clause in FLWR (which now becomes FLWOR)

16

17

- Example: list all books with price higher than \$100, in order by first author; for books with the same first author, order by title
 - <result>{
 - for \$b in doc("bib.xml")//book[@price>100]
 stable order by \$b/author[1], \$b/title empty least
 return \$b

}</result>

Summary

- Many, many more features not covered in class
- * XPath is fairly mature and stable
 - 1.0 is already a W3C recommendation
 - Implemented in many systems
 - Used in many other standards
 - 2.0 is being developed jointly with XQuery
- * XQuery is still evolving
 - Still a W3C working draft
 - Some vendors are coming out with implementations
 - To become the SQL for XML?
 - XQuery versus SQL
 - Where did the join go?
 - Strong ordering constraints (can be overridden by unordered { for... })