Introduction to Database Systems  
CSE 444

Lecture 12  
Xquery in SQL Server

October 22, 2007

Announcements

• Project phase 2 due Saturday night

• Midterm Next Monday. To study (tentative)
  – SQL
  – E/R diagrams
  – Functional dependencies and BCNF
  – XML Basics (notation, DTDs)
  • But hw3 is xquery/xpath, so most XML details on final
• Open book exam !!

Sorting in XQuery

<publisher_list>
{ FOR $b IN document("bib.xml")//book[year = "97"]
ORDER BY $b/price/text() 
RETURN <book>
  { $b/title , $b/price }
</book>
}
</publisher_list>

If-Then-Else

FOR $h IN //holding
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 }
Duplicate Elimination

- distinct-values(list-of-text-values)
- How do we eliminate duplicate “tuples”?

FOR vs. LET

FOR
- Binds node variables → iteration

LET
- Binds collection variables → one value

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...)

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...)

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

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

Summary:
- FOR-LET-WHERE-RETURN = FLWR

FOR/LET Clauses

List of tuples
WHERE Clause
List of tuples
RETURN Clause
Instance of Xquery data model

What about collections in expressions?
- $b/price$ → list of n prices
- $b/price * 0.7$ → list of n numbers
- $b/price * Sh/quantity$ → list of $n \times m$ numbers ??
- $b/price * (Sh/quant1 + Sh/quant2) ≠ b/price * Sh/quant1 + b/price * Sh/quant2$ !!!
Other XML Topics

- Name spaces
- XML API:
  - DOM = "Document Object Model"
- XML languages:
  - XSLT
- XML Schema
- Xlink, XPointer
- SOAP

Available from www.w3.org
(but don’t spend rest of your life reading those standards !)

XML in SQL Server 2005

- Create tables with attributes of type XML
- Use Xquery in SQL queries
- Rest of the slides are from:
  Shankar Pal et al., *Indexing XML data stored in a relational database*, VLDB’2004

XML Methods in SQL

- Query() = returns XML data type
- Value() = extracts scalar values
- Exist() = checks conditions on XML nodes
- Nodes() = returns a rowset of XML nodes that the Xquery expression evaluates to

Examples

- From here:

XML Type

CREATE TABLE docs (pk INT PRIMARY KEY, xCol XML not null)
Inserting an XML Value

```sql
INSERT INTO docs VALUES (2,
   '<doc id="123">
   <sections>
   <section num="1"><title>XML Schema</title></section>
   <section num="3"><title>Benefits</title></section>
   <section num="4"><title>Features</title></section>
   </sections>
   </doc>')
```

Query( )

```sql
SELECT pk, xCol.query('./doc[@id = 123]/section')
FROM docs
```

Exists( )

```sql
SELECT xCol.query('./doc[@id = 123]/section')
FROM docs
WHERE xCol.exist('./doc[@id = 123]') = 1
```

Value( )

```sql
SELECT xCol.value('data((/doc//section[@num = 3]/title)[1])', 'nvarchar(max)')
FROM docs
```

Nodes( )

```sql
SELECT nref.value('first-name[1]', 'nvarchar(50)') AS FirstName,
       nref.value('last-name[1]', 'nvarchar(50)') AS LastName
FROM @xVar.nodes('//author') AS R(nref)
WHERE nref.exist('first-name != "David"') = 1
```

Nodes( )

```sql
SELECT nref.value('@genre', 'varchar(max)') AS LastName
FROM docs CROSS APPLY
     xCol.nodes('//book') AS R(nref)
```
Internal Storage

• XML is “shredded” as a table
• A few important ideas:
  – Dewey decimal numbering of nodes; store in clustered B-tree index
  – Use only odd numbers to allow insertions
  – Reverse PATH-ID encoding, for efficient processing of postfix expressions like //a/b/c
  – Add more indexes, e.g. on data values

<BOOK ISBN="1-55860-438-3">
  <SECTION>
    <TITLE>Bad Bugs</TITLE>
    Nobody loves bad bugs.
    <FIGURE CAPTION="Sample bug"/>
  </SECTION>

  <SECTION>
    <TITLE>Tree Frogs</TITLE>
    All right-thinking people <BOLD>love</BOLD> tree frogs.
  </SECTION>
</BOOK>

Infoset Table

```
<table>
<thead>
<tr>
<th>ORDPATH</th>
<th>TAG</th>
<th>NODE_TYPE</th>
<th>VALUE</th>
<th>PATH_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(BOOK)</td>
<td>(Element)</td>
<td>Null</td>
<td>21</td>
</tr>
<tr>
<td>1.3</td>
<td>(SECTION)</td>
<td>(Element)</td>
<td>ISBN</td>
<td>1-55860-438-3</td>
</tr>
<tr>
<td>1.5.1</td>
<td>(TITLE)</td>
<td>(Element)</td>
<td>Bad Bugs</td>
<td>#453#1</td>
</tr>
<tr>
<td>1.5.3</td>
<td>(TEXT)</td>
<td>(Value)</td>
<td>&quot;Nobody loves bad bugs.&quot;</td>
<td>#108#1</td>
</tr>
<tr>
<td>1.5.4.5</td>
<td>(FIGURE)</td>
<td>(Element)</td>
<td>null</td>
<td>#454#1</td>
</tr>
<tr>
<td>1.5.5.1</td>
<td>(CAPTION)</td>
<td>(Attribute)</td>
<td>&quot;Sample bug&quot;</td>
<td>#653#1</td>
</tr>
<tr>
<td>1.5</td>
<td>(SECTION)</td>
<td>(Element)</td>
<td>null</td>
<td>#10#1</td>
</tr>
<tr>
<td>1.5.4</td>
<td>(TITLE)</td>
<td>(Element)</td>
<td>Tree frogs</td>
<td>#453#1</td>
</tr>
<tr>
<td>1.5.3</td>
<td>(TEXT)</td>
<td>(Value)</td>
<td>&quot;All right-thinking people love tree frogs.&quot;</td>
<td>#108#1</td>
</tr>
<tr>
<td>1.5.5</td>
<td>(BOLD)</td>
<td>(Element)</td>
<td>love</td>
<td>#783#1</td>
</tr>
<tr>
<td>1.5.7</td>
<td>(TEXT)</td>
<td>(Value)</td>
<td>tree frogs</td>
<td>#108#1</td>
</tr>
</tbody>
</table>
```