Lecture 12:
XQuery in SQL Server

Friday, April 20, 2007

Announcements

- Homework 2: deadline extended until tonight, at 11:59pm

- Midterm on Wednesday. To study:
  - SQL
  - E/R diagrams
  - Functional dependencies and BCNF

- Open book exam !!

- Project phase 2 due next Friday

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
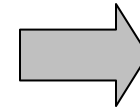
FOR $b IN //book

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" ?

```
<row>  <a>3</a>  <b>100</b> </row>
<row>  <a>8</a>  <b>500</b> </row>
<row>  <a>3</a>  <b>100</b> </row>
<row>  <a>3</a>  <b>200</b> </row>
<row>  <a>8</a>  <b>500</b> </row>
```

$\Rightarrow$

```
<row>  <a>3</a>  <b>100</b> </row>
<row>  <a>8</a>  <b>500</b> </row>
<row>  <a>3</a>  <b>200</b> </row>
```

FOR v.s. LET

FOR

- Binds *node variables* → iteration

LET

- Binds *collection variables* → one value

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>
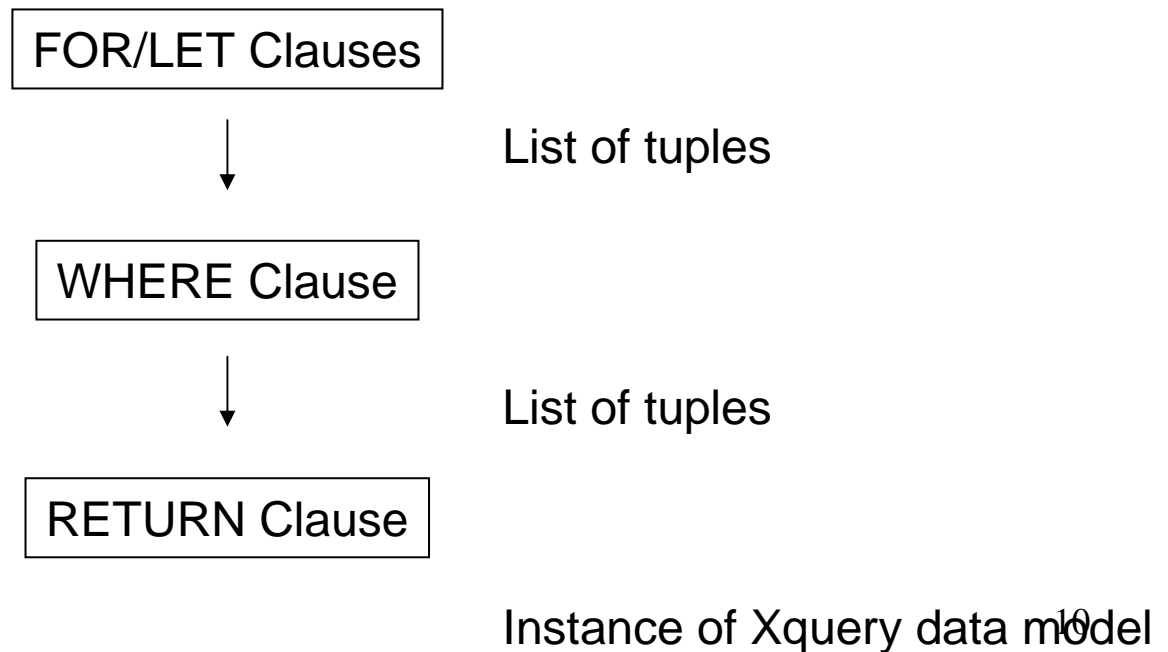   <book>...</book>
             <book>...</book>
             ...
    </result>

XQuery

Summary:

- FOR-LET-WHERE-RETURN $=$ FLWR

FOR/LET Clauses

↓ List of tuples

WHERE Clause

↓ List of tuples

RETURN Clause

Instance of Xquery data model

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>

11

Collections in XQuery

What about collections in expressions ?

- $b/price   → list of n prices

- $b/price * 0.7   → list of n numbers

- $b/price * $b/quantity → list of n x m numbers ??

- $b/price * ($b/quant1 + $b/quant2) ≠
   $b/price * $b/quant1 + $b/price * $b/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

```
CREATE TABLE DOCS (
    ID int primary key,
    XDOC xml)
```

```
SELECT ID, XDOC.query('
    for $s in  /BOOK[@ISBN= "1-55860-438-3"]//SECTION
    return <topic>{data($s/TITLE)} </topic>')
FROM DOCS
```

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:
  http://msdn.microsoft.com/library/default.as
  p?url=/library/en-
  us/dnsql90/html/sql2k5xml.asp

XML Type

```
CREATE TABLE docs (
    pk INT PRIMARY KEY,
    xCol XML not null
)
```

Inserting an XML Value

```
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( )

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

Exists( )

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

Value( )

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

Nodes( )

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)
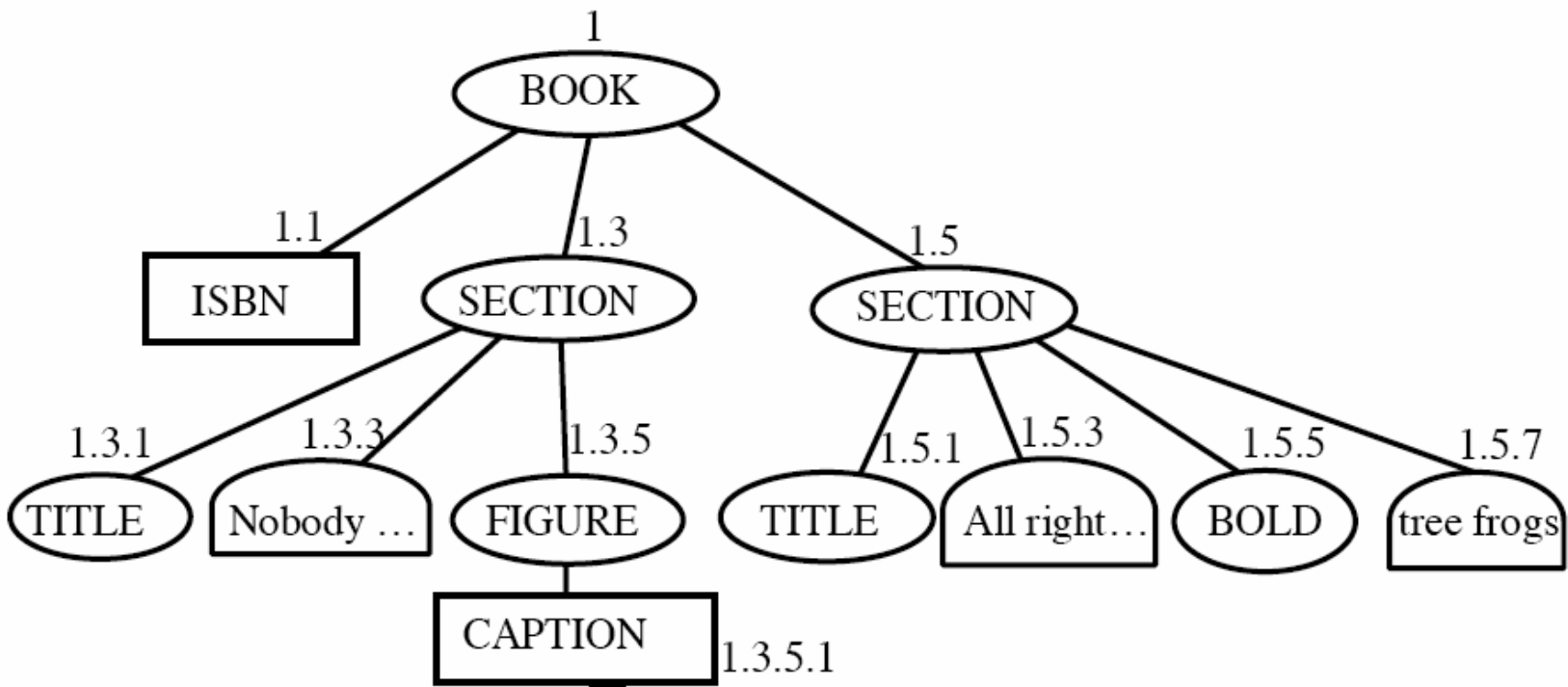WHERE nref.exist('.[first-name != "David"]') = 1

Nodes( )

```
SELECT nref.value('@genre', 'varchar(max)') 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 indes
  - 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

```xml
<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>
```
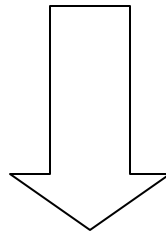
ORDPATH	TAG	NODE_TYPE	VALUE	PATH_ID
1	1 (BOOK)	1 (Element)	Null	#1
1.1	2 (ISBN)	2 (Attribute)	'1-55860-438-3'	#2#1
1.3	3 (SECTION)	1 (Element)	Null	#3#1
1.3.1	4 (TITLE)	1 (Element)	'Bad Bugs'	#4#3#1
1.3.3	10 (TEXT)	4 (Value)	'Nobody loves Bad bugs.'	#10#3#1
1.3.5	5 (FIGURE)	1 (Element)	Null	#5#3#1
1.3.5.1	6 (CAPTION)	2 (Attribute)	'Sample bug'	#6#3#1
1.5	3 (SECTION)	1 (Element)	Null	#3#1
1.5.1	4 (TITLE)	1 (Element)	'Tree frogs'	#4#3#1
1.5.3	10 (TEXT)	4 (Value)	'All right-thinking people'	#10#3#1
1.5.5	7 (BOLD)	1 (Element)	'love '	#7#3#1
1.5.7	10 (TEXT)	4 (Value)	'tree frogs'	#10#3#1

Infoset Table

/BOOK[@ISBN = "1-55860-438-3"]/SECTION

SELECT SerializeXML (N2.ID, N2.ORDPATH)
FROM infosettab N1 JOIN infosettab N2 ON (N1.ID = N2.ID)
WHERE N1.PATH_ID = PATH_ID(/BOOK/@ISBN)
    AND N1.VALUE = '1-55860-438-3'
    AND N2.PATH_ID = PATH_ID(BOOK/SECTION)
    AND Parent (N1.ORDPATH) = Parent (N2.ORDPATH)