Admin matters

- updated study guide
- essay presentations: 18 August (Topics 1, 2), 19 August (Topics 3, 4)

<table>
<thead>
<tr>
<th>Week</th>
<th>Date</th>
<th>Day</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>11 Aug</td>
<td>Tues</td>
<td>L7: Object-Relational DBMS</td>
</tr>
<tr>
<td></td>
<td>12 Aug</td>
<td>Wed</td>
<td>L8: Semi-structured databases</td>
</tr>
<tr>
<td></td>
<td>14 Aug</td>
<td>Fri</td>
<td><strong>Practical 3:</strong> Object-Relational DBMS (PostgreSQL)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Class Test 1:</strong> OODB and ORDB</td>
</tr>
<tr>
<td></td>
<td>18 Aug</td>
<td>Tues</td>
<td>L9:  Semi-structured databases</td>
</tr>
<tr>
<td></td>
<td>19 Aug</td>
<td>Wed</td>
<td>L10: Semi-structured databases</td>
</tr>
<tr>
<td></td>
<td>21 Aug</td>
<td>Fri</td>
<td><strong>Practical 4:</strong> XML DBs (BaseX)</td>
</tr>
</tbody>
</table>
In this lecture

- Revision (COS216 and INF272)
  - HTML and the web
  - XML, XML Schema, XML related technologies

- Reading for the student
  - Section 31.2 Introduction to XML
  - Section 31.3 XML-related technologies

- To be discussed in this lecture:
  - XML query languages
    - XQuery
      - XPath expressions
      - FLWOR queries
Semi-structured data

e.g. XML documents

- may have a structure, but may change

- aka schema-less or self-describing
  - no separate schema
  - schema is part of the data

- relational, object-oriented, object-relational DBMSs
  - Require pre-defined schema

```xml
<STAFFLIST>
  <STAFF branchNo = “B005”>
    <STAFFNO>SL21</STAFFNO>
    <NAME>
      <FNAME>John</FNAME><LNAME>White</LNAME>
    </NAME>
    <POSITION>Manager</POSITION>
    <DOB>1-Oct-45</DOB>
    <SALARY>30000</SALARY>
  </STAFF>
  <STAFF branchNo = “B003”>
    <STAFFNO>SG37</STAFFNO>
    <NAME>
      <FNAME>Ann</FNAME><LNAME>Beech</LNAME>
    </NAME>
    <POSITION>Assistant</POSITION>
    <SALARY>12000</SALARY>
  </STAFF>
</STAFFLIST>
```
Importance of semi-structured data

- **Recent importance**
  - treat Web resources as database
    - cannot constrain with schema
  - flexible format for data exchange
    - many different databases
  - becoming the **standard for data representation and data exchange** e.g. on the Web

- 1998: XML 1.0 by W3C
- Meta-language (language for describing other languages)
  - enables designers to create their **own customized tags**
  - provide **functionality not available with HTML**
XML: what you already know

1. XML file: e.g. myfile.xml

2. format of an XML document
   - header, tags, attributes, namespaces, etc

3. Document Object Model (DOM)
   - tree structure for an xml document

4. XML document validation
   - Document Type Definitions (DTDs)
   - Schemas (Microsoft XML schema, W3C schema)

5. Conversion of XML data to formatted text documents
   - Extensible Stylesheet Language (XSL, XSLT)
31.3.4 XML Path Language (XPath)

- Declarative query language
- Specifying a directory-like path
  - Possibly with conditions placed in the path
  - Retrieves collections of elements
- Treats XML document as logical tree of nodes

<table>
<thead>
<tr>
<th>Query component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>context node</td>
<td>starting point</td>
</tr>
<tr>
<td>location path</td>
<td>path from one point in document to another, composed of steps</td>
</tr>
<tr>
<td>step</td>
<td>consists of basis and predicates</td>
</tr>
<tr>
<td>basis</td>
<td>axis and node test</td>
</tr>
<tr>
<td>axis</td>
<td>direction e.g. parent, ancestor, etc.</td>
</tr>
<tr>
<td>node test</td>
<td>identifies node type, e.g. element name or function text()</td>
</tr>
<tr>
<td>predicate</td>
<td>in square brackets after the basis</td>
</tr>
</tbody>
</table>
XPath: path expressions

Details of all staff? \(/\) //STAFFLIST/child::*:STAFF or \(/\) //STAFFLIST/STAFF

Part of the query result:

```xml
<STAFF branchNo = “B005”>
  <STAFFNO>SL21</STAFFNO>
  <NAME>
    <FNAME>John</FNAME><LNAME>White</LNAME>
  </NAME>
  <POSITION>Manager</POSITION>
  <DOB>1-Oct-45</DOB>
  <SALARY>30000</SALARY>
</STAFF>
```
XPath: path expressions

Details of 1\textsuperscript{st} staff?

/child::STAFF[1] \textit{or} /STAFF[1] \textit{or} /child::STAFF[position()=1]
XPath: path expressions

name of 1st staff?

//STAFFLIST/STAFF[1]/NAME
XML Query Languages

- XML Query language
  - SQL not good for XML (irregularity of XML data)
  - XML-QL, UnQL, XQL, XQuery, ...

- XQuery by W3C XML Query Working Group
  - Contributions from
    - database community
    - document community
    - programming language community
XQuery

- XQuery
  - a functional language
  - a query is represented as an expression
  - nested expressions possible

- XQuery uses
  1. Path expressions (use the syntax of XPath)
  2. FLWOR expressions

- XML Queries
  - operate on single documents
  - operate also on fixed collections of documents
  - select sub-trees of documents
BaseX

BaseX folder:

```
<STAFFLIST>
  <STAFF branchNo="B005">
    <STAFFNO>SL21</STAFFNO>
    <NAME>
      <FNAME>John</FNAME>
      <LNAME>White</LNAME>
    </NAME>
    <POSITION>Manager</POSITION>
    <DOB>1965-10-01</DOB>
    <SALARY>30000</SALARY>
  </STAFF>
  <STAFF branchNo="B003">
```

```
.loggs
address
artifacts
Musicians
staff_list
```

```
TestDB
nok.xml
staff_list.xml
```
find staff number of first member of staff in our XML document

```
<STAFFNO>SL21</STAFFNO>
```

1. Opens `staff_list.xml` and returns its document node
2. uses `//STAFFLIST` to select STAFFLIST element
3. locates first STAFF element that is child of STAFFLIST element
4. finds STAFFNO elements occurring anywhere within this STAFF element
To extract the surnames of staff at branch B005, you can use the following XPath expression:

```
//STAFFLIST/STAFF[@branchNo="B005"]/NAME/LNAME/text()
```

Here's a breakdown of the XPath expression:

1. If `staff_list.xml` is the currently open database file, no need to specify the document.
2. Use `//STAFFLIST` to select the `STAFFLIST` element.
3. Use `/STAFF` to select `STAFF` elements within the `STAFFLIST` element.
4. Predicate: restricts `STAFF` elements to those with `branchNo` attribute = `B005`.
5. Selects `LNAME` element in `NAME` & extracts text.
**FLWOR (“flower”) expression construction:**

(1) starts with one or more **FOR** or **LET** clauses (any order)

**followed by:**

(2) optional **WHERE** clause

(3) optional **ORDER BY** clause

(4) required **RETURN** clause

**FOR** forVar **IN** inExpression

**LET** letVar := letExpression

**[WHERE** filterExpression **]**

**[ORDER BY** orderSpec **]**

**RETURN** expression
FLWOR Expressions

- **FOR** e.g. for $S$ in //STAFF
  - iteration: associates each variable with expression
  - result is tuple stream
  - each tuple binds a variable to one of the items in the expression

- **LET** e.g. let $\text{sal} := 30000$
  - binds one or more variables to one or more expressions
  - without iteration (single binding for each variable)

FOR forVar IN inExpression
LET letVar := letExpression
[WHERE filterExpression]
[ORDER BY orderSpec]
RETURN expression
FLWOR Expressions

• WHERE  e.g. where $S/SALARY > 10000

• one or more conditions to restrict tuples generated by FOR and LET (optional)

```plaintext
FOR  forVar IN inExpression
LET  letVar := letExpression
[WHERE  filterExpression]
[ORDER BY  orderSpec]
RETURN  expression
```
FLWOR Expressions

- **ORDER BY**  
  e.g. order by $S/STAFFNO$ descending
  - order of the tuple stream
  - determines order in which RETURN clause is evaluated

- **RETURN**  
  e.g. return $S$
  - evaluated once for each tuple in tuple stream
  - *query results concatenated to form returned result*

FOR forVar IN inExpression
LET letVar := letExpression
[WHERE filterExpression]
[ORDER BY orderSpec]
RETURN expression
List the names of all staff

```
for $N in doc("staff_list.xml")//STAFF/NAME
return $N
```

OR:
```
for $N in //STAFF/NAME  return $N
```
Example 31.4 – XQuery FLWOR Expressions

List staff with salary = R30,000

let $SAL := 30000
return doc("staff_list.xml")//STAFF[SALARY = $SAL]
(ALTERNATIVELY:
let $SAL := 30000
return //STAFF[SALARY = $sal] )

= extracts typed value of element (based on schema)
  • resulting in a decimal value
  • then compared with 30000
  • other operators
Example 31.4 – XQuery FLWOR Expressions

Find staff number at branch B005 with salary > R15,000

for $S in doc("staff_list.xml")//STAFF
where $S/SALARY > 15000 and $S/@branchNo = "B005"
return $S/STAFFNO

ALTERNATIVELY:
for $S in //STAFF
where $S/SALARY > 15000 and $S/@branchNo = "B005"
return $S/STAFFNO

<STAFFNO>SL21</STAFFNO>
Example 31.4 – XQuery FLWOR Expressions

List staff numbers of all staff in descending order of staff number

```
for $S in doc("staff_list.xml")//STAFF
order by $S/STAFFNO descending
return $S/STAFFNO
```

OR:
```
for $S in //STAFF
order by $S/STAFFNO descending return $S/STAFFNO
```
List each branch office and average salary at branch

```xquery
for $B in distinct-values(//@branchNo)
let $avgSalary :=
    avg(//STAFF[@branchNo = $B]/SALARY)
return
    <BRANCH>
    <BRANCHNO>{ $B }</BRANCHNO>
    <AVGSALARY>{ $avgSalary }</AVGSALARY>
</BRANCH>
```

Functions for aggregates are:
avg,..........................

Example 31.4 – XQuery FLWOR Expressions
Example 31.4 – XQuery FLWOR Expressions

List branches with at least one member of staff with salary > R15,000

for $B in distinct-values(//@branchNo)
let $S := //STAFF[@branchNo = $B]
where some $sal in $S/SALARY
  satisfies ($sal > 15000)
return <BRANCHNO>{ $B}</BRANCHNO>
Example 31.5 – Joining Two Documents

List names of staff and names of their next of kin

for $S$ in doc("data/staff_list.xml")//STAFF, $NOK$ in doc("data/nok.xml")//NOK
where $S$/STAFFNO = $NOK$/STAFFNO
return
  <STAFFNOK>
    { $S$/NAME, $NOK$/NAME } 
  </STAFFNOK>
Example 31.5 – Joining Two Documents

List names of all staff and names of their next of kin

```
for $S in doc("data/staff_list.xml")//STAFF
return <STAFFNOK>
    { $S/NAME } 
    { for $NOK in doc("data/nok.xml")//NOK
      where $S/STAFFNO = $NOK/STAFFNO
      return $NOK/NAME } 
</STAFFNOK>
```

Correlated sub-query
Example 31.6 – User-Defined Function

Simple arithmetic function in BaseX

FUNCTION DEFINITION:

let $\text{mult} := \text{function}(\$x, \$y) \{ \$x \times \$y \}$

FUNCTION CALL:

return $\text{mult}(10, 10)$

RESULT: ???

Built-in functions: Reading for the student
Example 31.6 – User-Defined Function – BaseX?

**Function to return staff at a given branch**

```xml
define function staffAtBranch($bNo) AS element()*{
    for $S in doc("data/staff_list.xml")//@STAFF
        where $S/@branchNo = $bNo
        order by $S/STAFFNO
        return $S/STAFFNO, $S/NAME, $S/POSITION,
        $S/SALARY
    }

for $B in
    distinct-values(doc("data/staff_list.xml")//@branchNo)
order by $B
return
  <BRANCHNO> { $B/text() } { staffAtBranch($B) } **the function call**</BRANCHNO>
```

Exercise for student: check if this works in BaseX
Tutorials on BaseX and XPath/XQuery

• The BaseX application is already installed in the labs. It can also be downloaded from http://www.basex.org.

• You will also find documentation and a demo section at the URL.

• Review the documentation about XPath/XQuery queries, download the demos provided and make sure you understand them.

• More information on XPath and XQuery are available at http://www.w3schools.com.