Admin matters

- 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>5</td>
<td>18 Aug</td>
<td>Tues</td>
<td>L9: Class Test 1: OODB and ORDB Presentation: essay topics 1 &amp; 2</td>
</tr>
<tr>
<td></td>
<td>19 Aug</td>
<td>Wed</td>
<td><strong>L10: Semi-structured: XML &amp; SQL</strong> Presentation: essay topics 3 &amp; 4</td>
</tr>
<tr>
<td></td>
<td>21 Aug</td>
<td>Fri</td>
<td><strong>Practical 4: XML DBs (BaseX)</strong></td>
</tr>
<tr>
<td>6</td>
<td>25 Aug</td>
<td>Tues</td>
<td>L11: Big Data and MongoDB</td>
</tr>
<tr>
<td></td>
<td>26 Aug</td>
<td>Wed</td>
<td>L12: MongoDB document database</td>
</tr>
<tr>
<td></td>
<td>28 Aug</td>
<td>Fri</td>
<td><strong>Practical 5: XML data and PostreSQL DBMS</strong></td>
</tr>
</tbody>
</table>
In this lecture

- Semi-structured data continued
- XML, databases and SQL
XML and Databases

Storing XML in a Relational or Object-Relational database

- Four general approaches:
  1. store XML as *value of some attribute* within a tuple
  2. store XML in a *shredded* form across a number of attributes and relations
  3. store XML in a *schema independent* form
  4. store the XML in a *parsed form* (internal)
Approach 1: Storing XML in an Attribute

<table>
<thead>
<tr>
<th>docNo</th>
<th>docDate</th>
<th>XMLdoc (whole doc or fragment)</th>
</tr>
</thead>
<tbody>
<tr>
<td>D001</td>
<td>2015-05-20</td>
<td><code>&lt;STAFFLIST&gt;</code>&lt;br&gt;<code>&lt;STAFF branchNo = &quot;B005&quot;&gt;</code>&lt;br&gt;<code>...</code>&lt;br&gt;<code>&lt;/STAFF&gt;</code>&lt;br&gt;<code>&lt;/STAFFLIST&gt;</code></td>
</tr>
</tbody>
</table>

- **Data types**
  - CLOB (character large object) – used in the past
  - Native XML data type: xml (PostgreSQL) or XMLType (Oracle)
- **Raw XML stored in the attribute**
  - efficient to *insert documents* into database
  - efficient to *retrieve documents*
  - easy to apply full-text *indexing* to documents
- **updates**: entire XML document is replaced (more recently an XML doc can be updated using SQL)
  - general *query performance is poor* due to parsing on the fly
Approach 2: Storing XML in Shredded Form

XML decomposed (shredded) into constituent elements

- **data distributed over number of attributes**
- **in one or more relations**
- **easier to index values of individual elements**
- **need additional data relating to hierarchical nature of the XML**
  - to recompose original document
  - XML updates
- **Have to create appropriate database structure from schema:** relational or object-relational

(reading for the student: pg 1123)

<table>
<thead>
<tr>
<th>BRANCHNO</th>
<th>STAFFNO</th>
<th>NAME</th>
<th>FNAME</th>
<th>LNAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOO5</td>
<td>SL21</td>
<td>Adam</td>
<td>Eden</td>
<td></td>
</tr>
<tr>
<td>BOO3</td>
<td>SG37</td>
<td>Eve</td>
<td>Eden</td>
<td></td>
</tr>
</tbody>
</table>
Approach 3: Schema-Independent Representation

e.g. create a relation from the Document Object Model (DOM) for the xml document

```
<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>
</STAFFLIST>
```
XML and SQL


  - native XML data type: XML
  
  - set of XML/SQL operators for the type
  
  - set of XML/SQL functions
  
  - implicit set of mappings from relational data to XML

© Pearson Education Limited 1995, 2005
Creating Table using XML Type

```
CREATE TABLE XMLStaff (  
docNo CHAR(4) PRIMARY KEY,  
docDate DATE,  
staffData XML );

INSERT INTO XMLStaff VALUES ('D001',  
DATE '2004-12-01',  
XML('<STAFF branchNo = "B005">  
<STAFFNO>SL21</STAFFNO>  
<POSITION>Manager</POSITION>  
<DOB>1945-10-01</DOB>  
<SALARY>30000</SALARY> </STAFF>');

SELECT * FROM XMLStaff;
```
SQL/XML Operators

• **XMLELEMENT**

  • generates XML value with a single element as a child of its root item.

  • Element can have zero or more attributes specified in the **XMLATTRIBUTES** sub-clause.
### SQL/XML Functions

**XMLFOREST**
- generates XML value with a list of elements as children of a root item.

**XMLCONCAT**
- concatenates a list of XML values

**XMLPARSE**
- performs a non-validating parse of a character string to produce an XML value

**XMLROOT**
- creates an XML value by modifying the properties of the root item of another XML value

**XMLCOMMENT**
- generates an XML comment

**XMLPI**
- generates an XML processing instruction

**XMLSERIALIZE**
- generates a character or binary string from an XML value

**XMLAGG**
- aggregate function, to generate a forest of elements from a collection of elements
Using XML Operators

List all staff as XML elements with name and branch number as an attribute

CREATE TABLE Staff (staffNo CHAR(4), fName CHAR(10), lName CHAR(10), branchNo CHAR(4));

INSERT INTO staff VALUES ('SL21', 'John', 'Green', 'B005');
INSERT INTO staff VALUES ('SG5', 'Susan', 'Brown', 'B003');

SELECT staffNo, xmlelement (NAME staff, 
xmlattributes (branchNo AS "branchNo"), 
concat(fName,lName)) AS staffXMLCol FROM Staff;

<table>
<thead>
<tr>
<th>staffno</th>
<th>staffxmlcol</th>
</tr>
</thead>
<tbody>
<tr>
<td>SL21</td>
<td>&lt;staff branchNo=&quot;B005&quot;&gt;John Green&lt;/staff&gt;</td>
</tr>
<tr>
<td>SG5</td>
<td>&lt;staff branchNo=&quot;B003&quot;&gt;Susan Brown&lt;/staff&gt;</td>
</tr>
</tbody>
</table>
SQL/XML Mapping Functions

SQL/XML
- defines mapping from tables to XML documents
- mapping produces two types of XML documents:

1. mapped table data
   - e.g. PostgreSQL: `table_to_xml`

2. XML Schema describing the mapped table data
   - e.g. PostgreSQL: `table_to_xmlschema`

- maps SQL data type to closest match in XML Schema
Functions for converting tables to XML (PostgreSQL 9.2 documentation – pg 247)

Queries:

CREATE TABLE numbers( num1 int, num2 int);
INSERT INTO NUMBERS VALUES(1,2);
INSERT INTO NUMBERS VALUES(3,4);
INSERT INTO NUMBERS VALUES(5,6);

SELECT * FROM NUMBERS;

Results
**PostgreSQL and XML data (2)**

Queries to generate XML from table

```sql
SELECT table_to_xml('Numbers', false, false,'');  --OR:
SELECT query_to_xml('SELECT num1, num2 FROM Numbers', false,false,'');
```

Output (when `table_to_xml` is used):

```
<table_to_xml
xml

<numbers xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <row>
    <num1>1</num1>
    <num2>2</num2>
  </row>
  <row>
    <num1>3</num1>
    <num2>4</num2>
  </row>
  <row>
    <num1>5</num1>
    <num2>6</num2>
  </row>
</numbers>
```

Row count: 1
Generate XML schema from table

Query:

SELECT table_to_xmlschema('Numbers', false, false, '');  -- OR:
SELECT query_to_xmlschema('SELECT num1, num2 FROM Numbers', false, false, '');

Output when `table_to_xmlschema` is used (partial listing):

```
  <xsd:schema
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:simpleType name="INTEGER">
      <xsd:restriction base="xsd:int">
        <xsd:maxInclusive value="2147483647"/>
        <xsd:minInclusive value="-2147483648"/>
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType name="RowType.TestingXML.public.numbers">
      <xsd:sequence>
        <xsd:element name="num1" type="INTEGER" minOccurs="0"></xsd:element>
        <xsd:element name="num2" type="INTEGER" minOccurs="0"></xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:schema>
```
CREATE TABLE XMLStaff (  
docNo CHAR(4) PRIMARY KEY,  
docDate DATE, staffData XML );

INSERT INTO XMLStaff VALUES ('D001',  
DATE '2004-12-01',  
XML('<STAFF branchNo = "B005">  
   <STAFFNO>SL21</STAFFNO>  
   <POSITION>Manager</POSITION>  
   <DOB>1945-10-01</DOB>  
   <SALARY>30000</SALARY> </STAFF>');

SELECT * FROM XMLStaff;

<table>
<thead>
<tr>
<th>docno</th>
<th>docdate</th>
<th>staffdata</th>
</tr>
</thead>
<tbody>
<tr>
<td>D001</td>
<td>2004-12-01</td>
<td>&lt;STAFF branchNo = &quot;B005&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;STAFFNO&gt;SL21&lt;/STAFFNO&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;POSITION&gt;Manager&lt;/POSITION&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;DOB&gt;1945-10-01&lt;/DOB&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;SALARY&gt;30000&lt;/SALARY&gt; &lt;/STAFF&gt;</td>
</tr>
</tbody>
</table>
PostgreSQL and XML data (4)

SQL and XPath queries:
syntax is: \( xpath(\ xpath-expression,\ colname) \)

```sql
SELECT  docNo, docDate, 
        xpath ('/STAFF/@branchNo', staffData) AS BranchNumber, 
        xpath ('/STAFF/STAFFNO/text()', staffData) AS StaffNumber, 
        xpath ('/STAFF/POSITION/text()', staffData) AS Position, 
        xpath ('/STAFF/DOB/text()', staffData) AS DOB, 
        xpath ('/STAFF/SALARY/text()', staffData) AS Salary 
FROM XMLStaff;
```

Results:

<table>
<thead>
<tr>
<th>docNo</th>
<th>docDate</th>
<th>branchnumber</th>
<th>staffnumber</th>
<th>position</th>
<th>dob</th>
<th>salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>D001</td>
<td>2004-12-01</td>
<td>{B005}</td>
<td>{SL21}</td>
<td>{Manager}</td>
<td>{1945-10-01}</td>
<td>{30000}</td>
</tr>
</tbody>
</table>
PostgreSQL and XML data (5)

SQL, XPath queries & unnest function:

```sql
SELECT docNo, docDate,
    unnest(xpath ('/STAFF/@branchNo', staffData) ) AS BranchNumber,
    unnest(xpath ('/STAFF/STAFFNO/text()', staffData)) AS StaffNumber,
    unnest(xpath ('/STAFF/POSITION/text()', staffData) ) AS Position,
    unnest(xpath ('/STAFF/DOB/text()', staffData) ) AS DOB,
    unnest(xpath ('/STAFF/SALARY/text()', staffData) ) AS Salary
FROM XMLStaff;
```

Results:

<table>
<thead>
<tr>
<th>docno character(4)</th>
<th>docdate date</th>
<th>branchnumber xml</th>
<th>staffnumber xml</th>
<th>position xml</th>
<th>dob xml</th>
<th>salary xml</th>
</tr>
</thead>
<tbody>
<tr>
<td>D001</td>
<td>2004-12-01</td>
<td>B005</td>
<td>SL21</td>
<td>Manager</td>
<td>1945-10-01</td>
<td>30000</td>
</tr>
</tbody>
</table>
Reading for the student

- Section 31.7: XML in Oracle
Presentation of topics 3 & 4 by students