XML in databases.
Some XML-related standards (XLink, XPointer, XForms).

Patryk Czarnik

XML and Applications 2015/2016
Lecture 12 – 30.05.2016

XML support in databases – categorisation

- Classic (usually relational) database with XML support
  - logical structure – relations and references
  - additional XML-related features
  - used for application integration or storing XML data as part of larger data structures
- `Native' XML database
  - logical structure – collection of XML document trees
  - XQuery (or XPath) as native query language
  - natural XML-related features
  - used for storing XML data (or structural data easily mapped to XML tree)

XML support in relational databases

- Possible functionalities
  - import and export of data in XML format
  - special treatment of XML data stored in fields
  - XML validation as part of integrity constraints checking
  - XPath or XQuery for querying field contents
  - XSLT applied to query result
- How to store XML data?
  - whole document (fragment) stored in single field
  - split into prima factors
    - each XML node in separate field
    - tables structure reflects tree structure of XML

Example – XML support in Oracle database

http://www.oracle.com/xml

- Since Oracle 8i
  - details differ from version to version
- XML parsers
  - for database programming (PL/SQL)
  - or middleware programming (Java, C++)
- XML-SQL Utility
  - XML data import and export
- `XMLType` data type and XML-specific operations

XML-SQL Utility

- `getXML()` function – XML data export

```
SELECT xmlgen.getXML('select * from emp') FROM dual;
```

```
<rowset>
  <row id="1">
    <empno>10</empno>
    <name>Scott Tiger</name>
    <title>specialist</title>
  </row>
  ...
</rowset>
```

XML in Oracle DB – XMLType

- `XMLType` – special datatype:
  - to be stored as LOB or used for columns, variables, etc.
  - indexing XML content
  - XPath expressions
  - validation against XML Schema
  - XSLT
- Available functions:
  - `extract`, `extractValue`, `exisitsNode`, `transform`, `updateXML`, `XMLSequence`

XMLType applications – some examples

```
CREATE TABLE warehouses(
  warehouse_id NUMBER(4),
  warehouse_spec XMLTYPE,
  warehouse_name VARCHAR2(35),
  location_id NUMBER(4));
```

```
CREATE TABLE po_xtab of XMLType;
```

```
UPDATE po_xml_tab
SET poDoc =  UPDATEXML(poDoc,
  '/PO/CUSTNAME/text()', 'John');
```

```
INSERT INTO warehouses VALUES
(100, XMLType(
 '<Warehouse whNo="100">
   <Building>Owned</Building>
  </Warehouse>'), 'Tower Records', 1003);
```

```
SELECT e.poDoc.getClobval() AS poXML
   FROM po_xml_tab e
   WHERE e.poDoc.existsNode('/PO[PNAME = "po_2"]') = 1;
```

```
CREATE INDEX city_index ON po_xml_tab
     (poDoc.extract('//PONO/text()').getNumberVal());
```

Source: Oracle DB documentation

XML support in database engines

- Specification: ISO/IEC 9075-14:2011 *SQL/XML*
  new data type XML:
  - only well-formed XML documents allowed
  - parsing and serialisation
  - implementation may add XML-specific operations
- Substantial support
  - IBM DB2 (since v.9 – *pureXML*)
  - Oracle (since 8i)
  - Microsoft SQL Server (since v.2000)
  - Sybase ASE (since v.12)
- Minimal support
  - MySQL – XPath queries over text fields containing XML
  - PostgreSQL – as above plus XML datatype
    but with no special operations

Native XML database

- Logical layer
  - XML document as basic data entity
  - collections of documents build a database
  - XML schema (or equivalent) as structure definition
  - XQuery (or XPath) as “native” query language
- Physical layer – not necessarily “files with XML text“
- More than just a collection XML files:
  - transactions and concurrent access
  - security (access privileges etc.), versioning, replication, ...
  - API for data access and update
  - additional means of data access
    - e.g. REST-compliant HTTP server
  - indexing for efficient access to selected nodes

Standards for XML databases

- High level query languages:
  - XQuery – primary language for queries
    - versions 1.0 and 3.0 / 3.1 in use
  - XQL – former approach to make XML query language
  - XPath – poor stub for XQuery
- High level update languages:
  - XQuery Update Extension
  - XUpdate
- Programmer APIs
  depend additionally on programming language
  - XML Database API (XAPI)
  - XQJ (for Java, expected to become XML equivalent of JDBC)
  - vendor-specific APIs...

XML:DB

- Initiative for XML database interfaces specification
  - XML Database API (XAPI)
  - accessing XML databases from programs
  - resource collections (resource = XML document)
  - reading and writing documents via DOM or SAX
  - pluggable “services”; specified: XPath, transactions, operations on collections
  - last version: 2001
- XML Update Language (XUpdate)
  - XML application (format) for updating XML databases
  - inserting, updating and removing nodes
  - XPath used for node addressing
  - last version: 2000

XUpdate – example

- Example (from XUpdate documentation)

```
<?xml version="1.0"?>
<xupdate:modifications version="1.0"
   xmlns:xupdate="http://www.xmldb.org/xupdate">
   <xupdate:insert-after select="/addresses/address[1]" >
      <xupdate:element name="address">
         <xupdate:attribute name="id">2</xupdate:attribute>
         <fullname>Lars Martin</fullname>
         <born day='2' month='12' year='1974'/>
         <town>Leizig</town>
         <country>Germany</country>
      </xupdate:element>
   </xupdate:insert-after>

  ...
 </xupdate:modifications>
```

XML database products – overview

Product	Licence	Queries	XML:DB API
eXist	open source	XPath, XQuery	yes
BaseX	open source	XPath, XQuery	yes
MarkLogic	commercial		
Apache XIndice	open source	XPath	yes
Sedna	open source	XPath, XQuery	yes
Gemfire Enterprise	commercial	XQuery, OQL	yes
Tamino	commercial	XQuery, XPath	part

Source: Wikipedia and providers' websites

In addition:

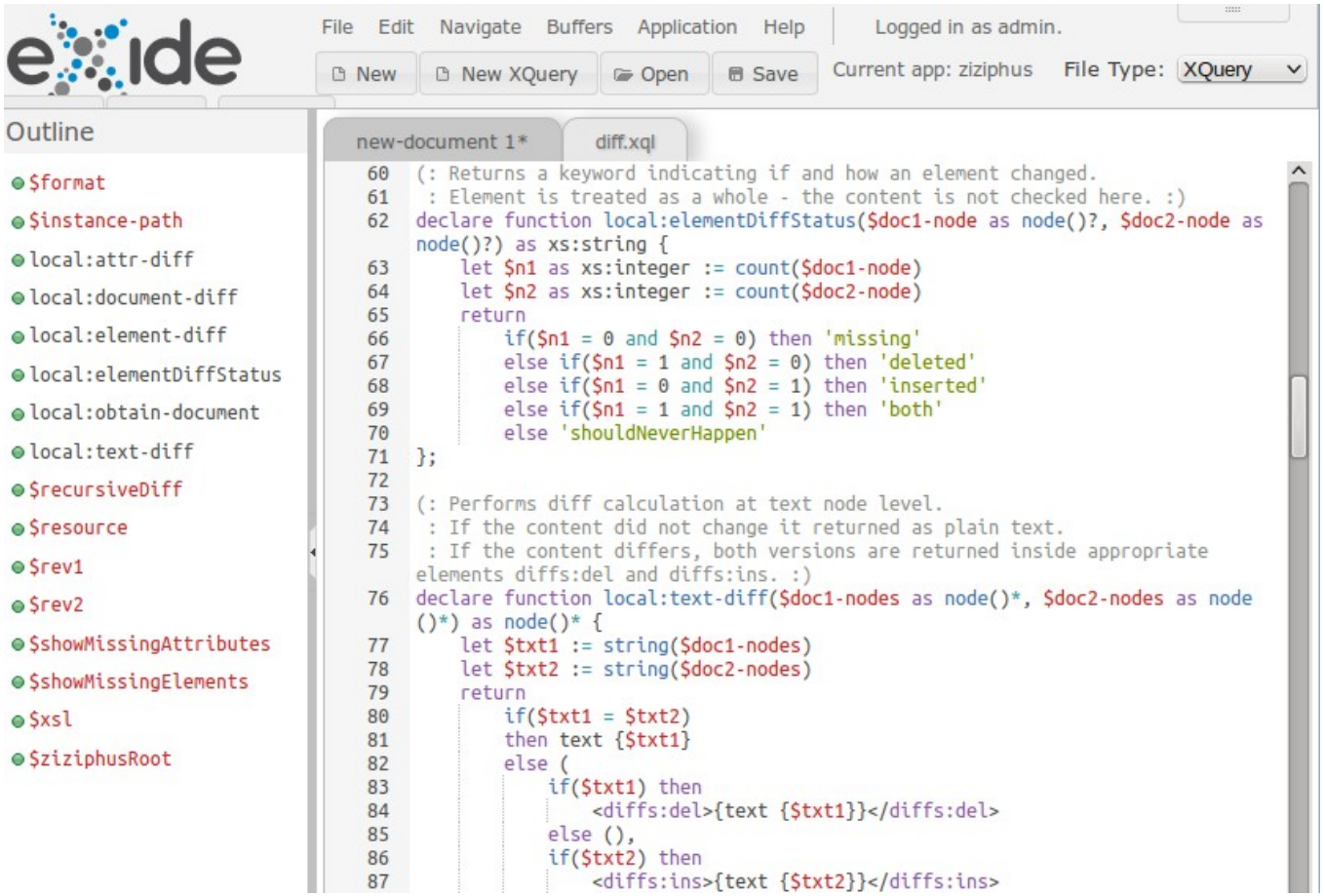
- Saxon – just a query processing engine, works on files (or other XML sources accessible in Java).

eXist DB

- One the most popular and elaborated XML database engines
- Open-source, but developed and supported by a (small German) company; commercial support available
- Features include:
  - storage of XML and binary entities
  - various means of access, including: human-readable Web interface, direct HTTP access (REST-compliant), SOAP and XML-RPC, Java API (XQJ, elements of XAPI)
  - full XML model available in XPath, XQuery, and XSLT code
  - full XQuery support with majority of new 3.1 features, Update extension and some other non-standard extensions
  - XForms support using betterFORM or XSLTForms plugins
  - extensible with custom Java code

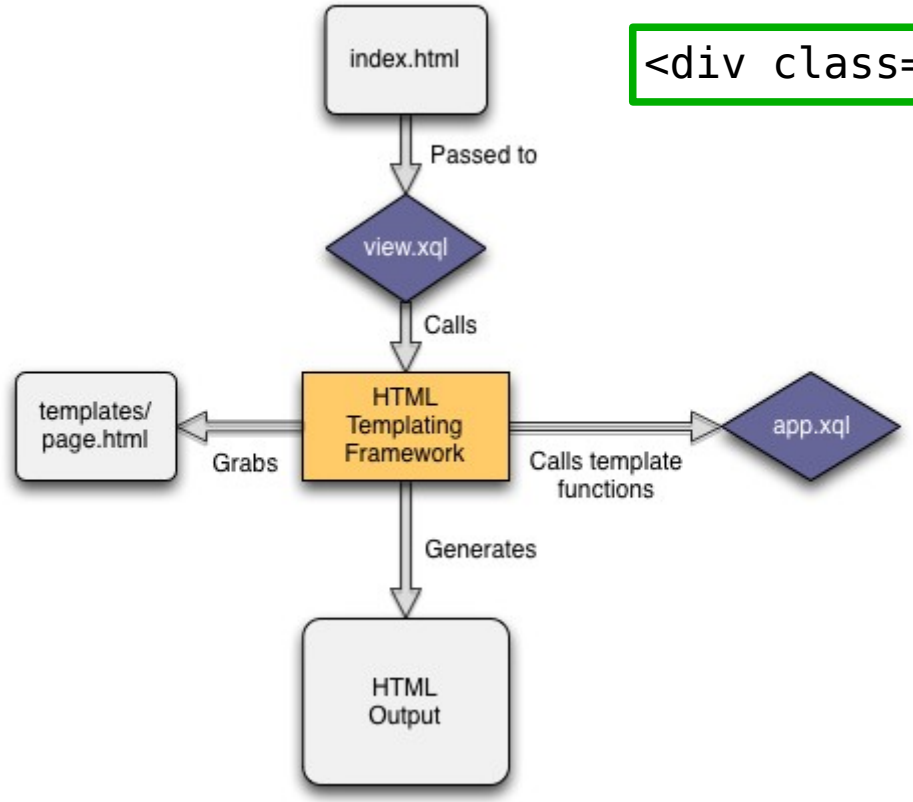
eXist – eXide

- XQuery programmer SDK running within a browser
  - supports also (to some extent...)
    XSLT, XML Schema, XHTML, XForms



eXist – template mechanism

- Easy integration of XQuery logic and HTML interface




```
<div class="app:hello"> </div>
```

```
declare function app:hello(
        $node as node(),
        $model as map(*), ...) {
    'Hello world!'
};
```

XForms

- XML application for interactive forms
- Versions:
  - 1.0 – 2003
  - 1.1 – 2009 (currently most commonly used)
  - 2.0 – neverending WD
- More than HTML forms:
  - data model defined separately from UI
    - by example or using XML Schema
  - processing model specified with events and actions
  - various data access modes given in submission module
    - including REST-compliant HTTP access
  - more UI controls, interactive switch, automatic repeat

XForms – document structure

- Forms are embedded in a host document, usually XHTML
- Data model – `xf:model` element
  - anywhere in host document
    - header more elegant
    - but body more practical for dynamic documents
  - more than one model available; in such case they must have identifiers
- Form controls (in XForms namespace)
  - placed within normal XHTML tags
  - (some of them) may contain further XHTML fragments
- Action specifications and constraints tied with XForm elements
  - by inserting them inside model fragments or control tags
  - using general `xf:bind` elements

Simple example

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:xf="http://www.w3.org/2002/xforms">
    <head>
        <xf:model>
            <xf:instance xmlns="">
                <person sex="">
                    <first-name/>
                    <last-name>Don</last-name>
                    <birth-date/>
                    <extra>
                        <position>assistant</position>
                        <salary cur="EUR">1500</salary>
                    </extra>
                </person>
            </xf:instance>
        </xf:model>
    </head>...
```

Simple example

```
<body>...
        <div>
            <xf:input ref="first-name">
                <xf:label>First name</xf:label>
                <xf:hint>Type your first (given) name</xf:hint>
            </xf:input>
...
            <xf:select1 ref="@sex">
                <xf:item>
                    <xf:label>woman</xf:label>
                    <xf:value>F</xf:value>
                </xf:item>
                <xf:item>
                    <xf:label>man</xf:label>
                    <xf:value>M</xf:value>
                </xf:item>
            </xf:select1>
            <xf:submit submission="submit">
                <xf:label>Submit</xf:label>
            </xf:submit>
        </div>
</body></html>
```

REST services – recall

- REST for Representational State Transfer
- Principles:
  - Service = collection of resources
  - URL identifies a resource
  - Resource has a normalised representation and can be transferred through the network
    - usually JSON or XML for structural data
    - binary and other formats also permitted, when appropriate
  - HTTP methods directly used to manipulate resources
    - GET, PUT, DELETE – obvious semantics
    - POST – creating new records or more complex operations
    - other HTTP methods, HTTP authentication, cookies, additional headers and arguments – all may be used to implement additional features

REST for XML database

- REST – remote access to a repository
  - Can it be an XML database? Why not...
- Possible applications:
  - Access API independent of particular platform or prog.lang.
  - Easy and efficient remote access from
    - Javascript clients (AJAX)
    - mobile clients
  - Integration with XML-related standards
    - XSLT, XQuery – documents available through HTTP URLs
    - XForms – acquiring and modifying documents directly form XForms
- HTTP interface available also to call server-side XQuery scripts
- XRX architecture: XForms + REST + XQuery

Standards for inter-document relations

- XPointer – addressing documents and their fragments
- XInclude – logical inclusion of documents within other documents
- XLink – declarative relations between documents and their fragments

XPointer

- The standard defines addressing XML documents and their fragments using standard URI syntax:
  - http://www.sejm.gov.pl/ustawa.xml#def-las
- 3 W3C recommendations dated 2002-2003:
  - XPointer Framework
    http://www.w3.org/TR/xptr-framework/
  - XPointer element() Scheme
    http://www.w3.org/TR/xptr-element/
  - XPointer xmlns() Scheme
    http://www.w3.org/TR/xptr-xmlns/
  - XPointer xpointer() Scheme
    http://www.w3.org/TR/xptr-xpointer/
    - (neverending?) Working Draft

XPointer – xpointer scheme

- xpointer scheme allows to address elements using XPath:
  - http://www.sejm.gov.pl/ustawa.xml#xpointer(/art[5]/par[2])
- xmlns scheme adds namespace declarations to the above:
  - ustawa.xml#xmlns(pr=http://www.sejm.gov.pl/prawo) xpointer(/pr:art[5]/pr:par[2])

XPointer – element scheme

- Element carrying ID attribute with given value:
  - document.xml#element(def-las)
- Element with given position (absolute or relative to element carrying ID with given value):
  - document.xml#element(/1/4/3)
  - document.xml#element(def-las/2/3)
- Short syntax:
  - document.xml#def-las
  - document.xml#/1/4/3
  - document.xml#def-las/2/3

XInclude

- Including external XML documents (or their fragments) in another XML document.
- Similar to entities, but:
  - normal element markup, no special syntax,
  - no need to declare anything in DTD, nor to have DTD at all
- Main capabilities:
  - including complete documents (identified by URL) or their fragments (pointed by XPointer)
  - including XML tree (default) or raw text
  - defining content to be used in case of an error
- Supported by many parsers, including Java (JAXP).

XInclude – example

```
<recipe>
   <xi:include xmlns:xi="http://www.w3.org/2001/XInclude"
      href="salad.xml#xpointer(/recipe/title)">

      <xi:fallback>
         <error>No such recipe.</error>
      </xi:fallback>
    </xi:include>
</recipe>
```

XLink

- HTML links (`<a>`, `<img>`):
  - link two documents: link source and target
  - link source is always in the linking element
- XLink — an extended idea of linking:
  - link information represented in any element:
    - element name is not important
    - attributes coming from XLink namespace are
  - more than two ends of link (hyperlink → relation)
  - possibility to represent link outside linked resources
- Status:
  - historical roots: HyTime,
  - XLink 1.0 – W3C recommendation: 2001,
  - XLink 1.1 – current version (made official TR: May 2010).

Terminology

- **Resource** – any addressable unit of information or a service (file, program, query result).
- **Link** – a relation between participating resources, expressed explicitly with a linking element.
- **Arc** – information about traversal between labelled resources (in defined direction):
  - outbound arc – from a local resource to some external resource
  - inbound arc – from an external resource to some local resource
  - third party – between two external resources
- Note: a resource is regarded as remote when addressed by URI (even though it resides in the same document or linking element as the link which uses it).

Types of links

- Simple link:
  - is outbound
  - binds exactly two resources: a local one with an external one
  - contains exactly one arc between resources
- Extended link:
  - binds arbitrary number of local and external resources,
  - uses arcs to define methods of traversal between resources,
  - defines roles of participating resources,
  - defines roles of arcs.

Simple link – an example

```
<book xmlns:xlink="http://www.w3.org/1999/xlink">
   <author xlink:type="simple"
        xlink:href="http://www.example.com/
                    bookstore/authors/Cormen">Thomas H. Cormen</author>
   <title>Introduction to algorithms</title>
</book>
```

Extended link – an example

```
<family xlink:type="extended" xmlns:xlink="http://www.w3.org/1999/xlink">
   <person xlink:type="locator" xlink:href="joe.xml"
           xlink:label="parent" xlink:title="Joseph"/>
   <person xlink:type="locator" xlink:href="cathy.xml"
           xlink:label="parent" xlink:title="Katherine"/>
   <person xlink:type="locator" xlink:href="mikey.xml"
           xlink:label="child" xlink:title="Michael"/>
   <person xlink:type="locator" xlink:href="toya.xml"
           xlink:label="child" xlink:title="La Toya"/>
   <person xlink:type="locator" xlink:href="janet.xml"
           xlink:label="child" xlink:title="Janet"/>
   <link xlink:type="arc" xlink:from="parent" xlink:to="child"/>
</family>
```

Attributes in extended links

- **type** – role of the element in a link
  - simple | extended | locator | arc | resource | title | none
- **href** – URI of the external resource
- **role** – abstract identifier of the resource role (URI)
- **arcrole** – as above, but for an arc
- **title** – text label of the resource or arc
- **show** – presentation info: new | replace | embed | other | none
- **actuate** – activation info: onLoad | onRequest | other | none
- **label** – label used as identifier in from and to, not necessarily unique
- **from**, **to** – pointer (in an arc) for a certain resource label

Future of XLink

- Applications:
  - organization and association of resources even when no writing permission is granted
  - a new type of added value – link sets
- Scope:
  - local – link servers, link databases
  - Internet?
- Problems:
  - visualization of extended links
  - synchronization of links and resources (Internet)