Prof. Dr.-Ing. Stefan Deßloch
AG Heterogene Informationssysteme
Geb. 36, Raum 329
Tel. 0631/205 3275
dessloch@informatik.uni-kl.de

Technische Universität Kaiserslautern

Chapter 10 - XML

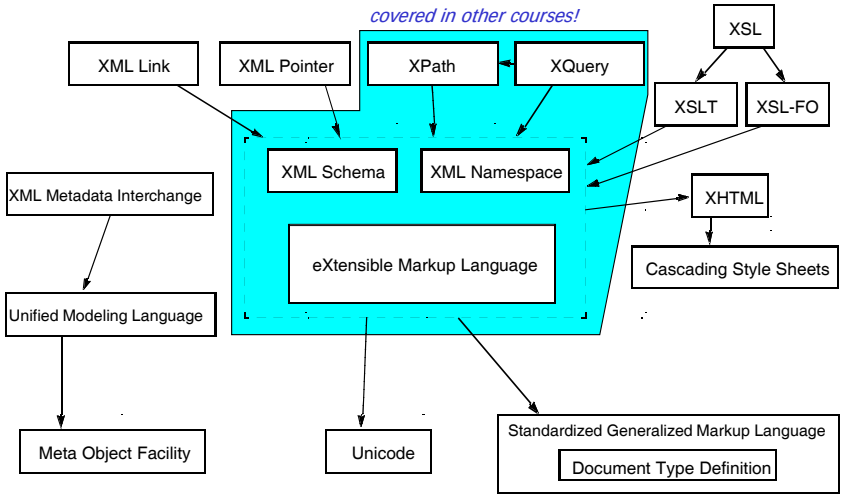
Forces Driving XML

- Document Processing
  - Goal: use document in various, evolving systems
  - structure – content – layout
  - grammar: markup vocabulary for mixed content
- Data Bases and Data Exchange
  - Goal: data independence
  - structured, typed data – schema-driven – integrity constraints
- Semi-structured Data and Information Integration
  - Goal: integrate autonomous data sources
  - data source schema not known in detail – schemata are dynamic
  - schema might be revealed through analysis only after data processing

XML and CM/DL

- XML can be used to represent documents and data
  - content and structure
- XML is a text-oriented language
  - text search supported in XQuery Full Text standard
  - not really suitable for multimedia content beyond text
- Multi-media content can be referenced in XML documents
  - URI, XLink, XPointer, XPath
  - Synchronized Multimedia Integration Language (SMIL)
- Multi-media content can be encoded in a text-based format
  - Scalable Vector Graphics (SVG)
- XML processing standards support flexible generation of different layout
  - XML Style Sheets Transformations
- XML for meta-data representation
  - RDF
  - Meta-data standards (e.g., Dublin Core)
    - Representation in XML

XML Language Specifications (W3C)



XQuery Full-Text (XQFT) Extensions

- XQuery
  - focuses on querying the structure of XML documents
  - provides only rudimentary support for querying text content
    - function fn:contains(<stringexpr1>, <stringexpr2>)
      returns true, iff <stringexpr1> contains the substring <stringexpr2>
  - Example:

```
for $b in /books/book
let $text = $b/fn:string()
where fn:contains($text, "web site")
  and fn:contains($text, "usability")
return <result>
            <title> {$b//title} </title>
       </result>
```

- XQFT
  - extends XQuery with text search/retrieval capabilities
    - ftcontains expression supports boolean full-text search
    - enhancements of FLWOR expressions to support scoring (ranking)

Boolean Full-Text Search (in one chart)

- Full-text search in general
  - perceives text not as a character string, but as a series of words/tokens
    - may recognize further well-defined units such as sentences, paragraphs
  - search identifies text in which tokens occur that match a search condition
    - goes beyond string equality, allows for variations
      - case insensitive, stemmed forms, sounds-like, fuzzy matching, regular expressions
      - may be language-sensitive
    - may employ measures of 'similarity' of a retrieved document with a search pattern or document
      - different retrieval models
      - quality of search capabilities defined by precision, recall measures
      - scoring/ranking of results
- Boolean FT Search
  - simple retrieval model based on set theory and boolean algebra
    - queries specified as boolean expressions
    - utilize basic search predicates for keyword/phrase search including numerous variations
    - may also involve proximity search to retieve documents where certain tokens appear
      - in the same sentence/paragraph
      - within a certain range of each other (e.g., at most 5 words apart)
  - variations for introducing scoring, generalizing the semantics of boolean search patterns

XQFT FTContains Expression

- Syntax:
  <range> **contains text** <FTSelection> [**without content** <ignore>]
  - <range>, <ignore> are expressions that define the scope of the full-text search
    - a sequence of nodes over which the search is performed
  - <FTSelection> specifies a full-text search condition to be evaluated over the scope
  - example: give me all books containing "XQuery FullText", ignoring footnotes
    /books/book[. contains text "XQuery FullText" without content .//footnote]/title
- FTSelections may again contain nested XQuery expressions
  - example: give me all books having at least one section containing all words in the book title
    /books/book[.//section contains text {title} all words]/title
- Nodes returned by the scope expressions are tokenized in an implementation-defined manner
  - usually based on the text content (string value) of an element
    - attributes, tags of nested elements may be ignored
  - returns a sequence of tokens (with positional information)
  - can exploit structure to group tokens into logical units (e.g., sentence, paragraph)

FTSelection Expressions

- Word and phrase matching (see previous examples)
  - additional options allow to specify whether to search for individual words or for phrases, and whether all or some words need to be found to have a successful match
- Boolean operators
  - or ("ftor"), and ("ftand"), not ("ftnot")
  - weak not: not in
    - example: find books about "Mexico", not "New Mexico"
      //book[. contains text "Mexico" not in "New Mexico"]/title
    - text “New Mexico is named after Mexico.” matches the above search condition
- Distance/proximity predicates
  - search for words appearing in the same/in a different sentence/paragraph
    - example:
      //book[. contains text ("web" ftand "site" ftand "usability") same sentence]/title
  - can involve maximum distance in terms of words, sentences, paragraphs
  - can be based on a (sliding) window
- Order of words, number of occurrences can be specified as well

Match Options

- Stemming/linguistic search (e.g., "use" vs. "used" vs. "using")
  - search for exact work appearance or word variations/inflections
- Character case variations
  - case insensitive (default) or sensitive, lowercase, uppercase
- Diacritics (e.g., "naïve" vs. "naive")
  - insensitive, sensitive, with, without
- Character wildcards
  - single, optional, zero or more, one or more, between n and m characters
- Thesaurus expansion (e.g., "canine" vs. "dog" vs. "poodle")
  - expands query terms based on relationships defined in a thesaurus
    - synonym, boader term, narrower term, related term, ...
  - thesaurus option can identify the thesaurus to use, the expansion relationship, and for hierarchical relationships the number of levels to expand
- Control which words are regarded as stopwords (e.g., "but", "if", ...)
  - without stopwords, or with the default or a specific stopword list
- Language used in documents or query

Scoring

- FTContainsExpr returns a boolean value
  - no indication about how well the search context nodes match the query
  - a node with a single occurrence of one of the search words is rated the same as a word with many occurrences of all the search words
- Score values reflect the relevance of the context nodes regarding the search
  - value in the range [0 1]
  - higher value (for value > 0) means higher relevance
- Numerous scoring algorithms have been proposed
  - standard approach involves term frequency (tf) and inverse document frequency (idf) measures, i.e., the score will be higher if
    - the number of matches for a search term is higher,
    - the search term matches fewer documents overall
  - XQFT does not prescribe a specific scoring algorithm
- A result "false" for ftcontains does not imply "score = 0", and vice versa
  - Example: "XML" ftand "FullText"
    - ftcontains returns false, if the context node does not contain both terms
    - score may be >0, if the node contains one of the terms

XQFT Score Variables

- Score variables
  - can be bound to the score values of full-text matches
  - are special variables introduced in the for/let-clauses of XQuery
  - no impact on binding of other variables
- Score variables in the for-clause
  - Example:

```
for $b score $s in /books/book[content contains text "web site" ftand "usability"]
where $s > 0.5
order by $s descending
return <result>
            <title> {$b//title} </title>
            <score> {$s} </score>
       </result>
```

  - for each item bound to a "regular" variable in the for clause (e.g., $b), the score is determined and bound to the score variable (e.g., $s)
    - "dual" purpose of the in-clause: filtering and scoring

XQFT Score Variables (cont.)

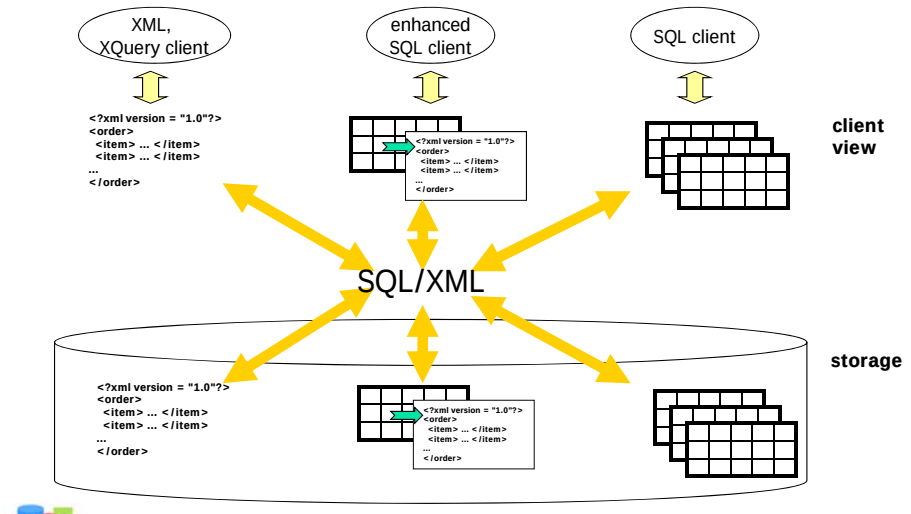
- Score variables in the let-clause
  - allows to separate filtering from scoring aspects
  - Example:

```
for $b in /books/book[.//chapter/title contains text "testing"]
let score $s := $b/content contains text "web site" ftand "usability"
order by $s descending
return <result score="{$s}">{$b}</result>
```

  - The above query performs scoring (in the let clause) on different match criteria than filtering (in the for clause)
- Score values may be fine-tuned using weights
  - allow to (de-)emphasize certain parts of the search condition
  - Example:

```
for $b in /books/book[.//chapter/title contains text "testing"]
let score $s := $b/content contains text ("web site" weight 0.2)
                                    ftand ("usability" weight 0.8)
order by $s descending
return <result score="{$s}">{$b}</result>
```

SQL/XML Big Picture



Hybrid SQL/XML Databases

- Increasing importance of XML in combination with data management
  - flexible exchange of relational data using XML
  - managing XML data and documents
  - trend towards "hybrid" approaches for relational DBMS
- SQL/XML standard attempts to support the following
  - "Publish" SQL query results as XML documents
  - Ability to store and retrieve (parts of) XML documents with SQL databases
  - Rules and functionality for mapping SQL constructs to and from corresponding XML concepts
- Relies partly on XQuery standard
  - XML data model
  - queries over XML data
- Broad support by major SQL DBMS vendors

XLink - XML Linking Language

- Hyperlinks, references in XML documents
  - separate specification
  - based on Uniform Resource Identifiers (URIs), XPath, XPointer as referencing mechanisms
  - more powerful than HTML hyperlinks (see chapter 7)
    - bi-directional, more than two resources (n-ary links)
    - powerful addressing of resources
      - direct reference to object components
    - link attributes provide metadata
    - storage of links independent of resources
  - but also provides support for "simple" links (comp. to HTML) through special abbreviated syntax

XLink Elements and Attributes

- XLink Element
  - has special attributes defined in XLink namespace
    - **type (simple, extended, ...)**
    - **href** (URI-reference or XPointer)
    - **title, role** (describes link semantics)
    - **show (new, replace, embed, undefined)** (activation behavior)
    - **actuate (onLoad, onRequest, undefined)** (link traversal)
    - **from, to** (definition of directed edges in link graph)
- Link types
  - outbound: local start resource, remote end resource
  - inbound: remote start resource, local end resource
  - third-party: remote start resource, remote end resource

XLink - Example

- in DTD:

```
<!ELEMENT Player ANY>
<!ATTLIST Player
                xlink:type  (simple)   #FIXED    "simple"
                xlink:href  CDATA   #REQUIRED
                xlink:role   NMTOKEN  #FIXED "http://www.fck.com/links/spieler"
                xlink:title   CDATA    #IMPLIED
                xlink:show    (new|embed|replace)   "replace"
                xlink:actuate (onLoad|onRequest)   "onRequest"
        >
```

- in document instance:

```
< Player xlink:href="http://www.fck.de/Spielerliste.xml"
                xlink:title="List of all FCK players"
                xlink:show="new">
                Here's a list of all FCK players.
</Player>
```


XSL – Transformation and Layout

- **XSL**: *Extensible Stylesheet Language*
  - formatting engine for XML
  - XML markup is presentation/layout-independent
- **XSLT**: *XSL Transformation Language*
  - stylesheet
  - transformation rules
    - consisting of a pattern and a template
    - usage of XPath
  - input tree
  - output tree
- **XSL-FO**: *XSL Formatting Objects*
  - vocabulary for the specification of formatting rules
    - reuse of complex formatting definitions
  - transformation into arbitrary formats (PDF, RTF, PostScript, ...)

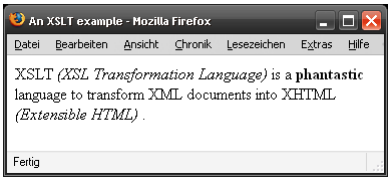
Principles of XSLT

- XSL processor
  - element-by-element processing of input tree, starting with the root
  - looks for applicable XSLT rule
- XSLT rule defines
  - template: for which element, in which relationship context does the rule apply
  - action:
    - what should be generated as output
      - reference to document content using 'select'-expressions
    - what elements should be processed next by the processor
      - <xsl:apply-templates/> - continue with child elements
      - extended syntax supports selection of specific elements, order restrictions, etc.
- Default rules (if no other rule is matched)
  - for all elements (incl. root), process the children
  - for all text nodes and attributes, use their value as output

XSL - Example

- a fragment of an XML document:

```
<Content>
<Paragraph>XSLT <foreignTerm> (XSL Transformation Language) </foreignTerm >
is a <Emphasis> phantastic</Emphasis> language to transform XML documents into
XHTML <foreignTerm> (Extensible HTML) </foreignTerm>.
</Paragraph>
</Content>
```



- result document:

```
<html>
  <head>
      <title>An XSLT example</title>
  </head>
  <body>
      XSLT <i>(XSL Transformation Language)</i>
       is a <b> phantastic </b> language to
      transform XML documents into XHTML
      <i> (Extensible HTML) </i>.
      </body>
</html>
```

XSL - Example

- XSLT-Stylesheet:

```
<xsl:stylesheet        xmlns:xsl=http://w3.org/XSL/Transform/1.0
                       xmlns=http://w3.org/TR/xhtml1
                       indent-rules="yes">
<!– Rule 1 -->         <xsl:template match="/">
                              <html>
                              <head>
                                     <title>An XSLT example</title>
                              </head>
                              <body>
                                     <xsl:apply-templates/>
                              </body>
                              </html>
                       </xsl:template>
<!-- Rule 2 -->       <xsl:template match="Paragraph">
                              <xsl:apply-templates/>
                       </xsl:template>
<!-- Rule 3 -->       <xsl:template match="Emphasis">
                              <b><xsl:apply-templates/></b>
                       </xsl:template>
<!-- Rule 4 -->       <xsl:template match="foreignTerm">
                              <i><xsl:apply-templates/></i>
                       </xsl:template>
```

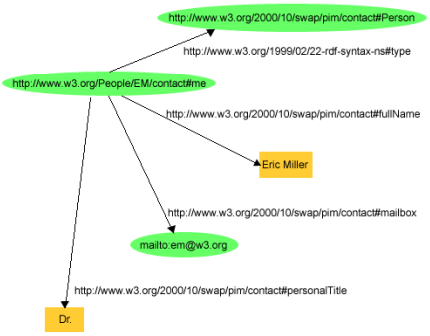

Resource Description Framework (RDF)

- Language for representing information (e.g., meta data) about resources on the web
  - identify something on the web using Uniform Resource Identifier (URI)
  - describe it using simple property/value pairs
- RDF statement can be represented using a graph
  - Example (from the RDF spec): "there is a Person identified by http://www.w3.org/People/EM/contact#me, whose name is Eric Miller, whose email address is em@w3.org, and whose title is Dr."
- RDF heritage: knowledge representation
  - semantic networks

RDF/XML

- XML-based syntax for encoding and exchanging RDF statements
  - Example

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf=...>
<contact:Person
  rdf:about=
  "http://www.w3.org/People/EM/contact#me">
   <contact:fullName>
                Eric Miller
   </contact:fullName>
   <contact:mailbox
     rdf:resource="mailto:em@w3.org"/>
  <contact:personalTitle>
                Dr.
  </contact:personalTitle>
</contact:Person>
</rdf:RDF>
```



- Could be used to provide semantic markup for XHTML documents
  - extension of the HTML META tag

Dublin Core

- Meta-data standard for describing networked resources
  - established by international, cross-disciplinary group of professionals from librarianship, computer science, text encoding, the museum community, and other related fields
  - major goal: help improve resource discovery
  - also used in closed environments, for other purposes(e.g., meta-data exchange)
- Meta-data description
  - uses a set of common meta data elements (nouns) and qualifiers (adjectives)
    - can be embedded in the resource (e.g., as HTML meta tags)
    - can be contained in a separate record/description of a resource (e.g., in a meta-data catalog file or database)
- Dublin Core defines
  - a vocabulary for meta data
    - simple to use, based on commonly used semantics, international, extensible,
  - best practices of how to use the language in various formats
    - HTML, XML, RDF

Dublin Core Elements

- Elements can be broadly grouped into three categories
  - Content
    - **Title**: A name given to the resource.
    - **Subject**: The topic of the content of the resource.
    - **Description**: An account of the content of the resource.
    - **Type**: The nature or genre of the content of the resource.
    - **Source**: A reference to a resource from which the present resource is derived.
    - **Relation**: A reference to a related resource.
    - **Coverage**: The extent or scope of the content of the resource.
  - Intellectual Property
    - **Creator**: An entity primarily responsible for making the content of the resource.
    - **Contributor**: An entity responsible for making contributions to the resource content.
    - **Publisher**: An entity responsible for making the resource available
    - **Rights**: Information about rights held in and over the resource.
  - Instantiation
    - **Date**: A date associated with an event in the life cycle of the resource.
    - **Format**: The physical or digital manifestation of the resource.
    - **Identifier**: An unambiguous reference to the resource within a given context.
    - **Language**: A language of the intellectual content of the resource.

Qualifiers

- Each element is optional and repeatable
- There is no defined order of elements
- Definition of controlled vocabularies possible (i.e., permitted values for elements)
  - uses concept of qualifiers
- Two broad classes
  - Element Refinement: make the meaning of an element narrower or more specific
  - Encoding Scheme: identify schemes that aid in the interpretation of an element value
- Example: Element **Date**
  - Refinements
    - Created, Valid, Available, Issued, Modified, Date Copyrighted, Date Submitted
  - Encoding Schemes
    - DCMI Period, W3C-DTF
- Example: Element **Relation**
  - Refinements
    - Is Version Of, Has Version, Is Replaced By, Replaces, Is Required By, Requires, Is Part Of, Has Part, Is Referenced By, References, Is Format Of, Has Format, Conforms To
  - Encoding Scheme
    - URI

DC XML Implementation Guidelines

- Each DC element is represented as a separate XML element
  - refinements become elements of their own
  - encoding schemes are represented using xsi:type
- Example

```
<metadata xmlns=...>
  <dc:title> UKOLN </dc:title>
  <dc:subject> national centre, network information support</ dc:subject>
  <dc:identifier xsi:type="dcterms:URI"> http://www.ukoln.ac.uk/ </dc:identifier>
  <dcterms:modified xsi:type="dcterms:W3CDTF">
        2001-07-18
  </dcterms:modified>
  ...
</metadata>
```


DC RDF Implementation Guidelines

- Each resource is described in an RDF description element
  - most appropriate URI to be used for rdf:about attribute
- Example

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:dc ="http://purl.org/dc/elements/1.1/">
  <rdf:Description rdf:about="http://dublincore.org/">
    <dc:title>Dublin Core Metadata Initiative - Home Page</dc:title>
    <dc:description>
        The Dublin Core Metadata Initiative Web site.
    </dc:description>
    <dc:date>2001-01-16</dc:date>
    <dc:format>text/html</dc:format>
    <dc:language>en</dc:language>
    <dc:contributor>The Dublin Core Metadata Initiative</dc:contributor>
  </rdf:Description>
</rdf:RDF>
```