

# XML as a Semistructured Data Model

Prof. Dr. Wolfgang May  
Universität Göttingen

## Overview

- What is “Semistructured Data”?
- Basic ideas and goals of XML?
- data model, representation ...

# Relational Data Model

- **database = schema + contents**
- schema/metadata: relation names, attributes
- contents: tuples

Country				
Name	<u>code</u>	Population	Capital	...
Germany	D	83536115	Berlin	..
Belgium	B	10170241	Brussels	..
Canada	CDN	28820671	Ottawa	..
..	..	..	..	..

XML

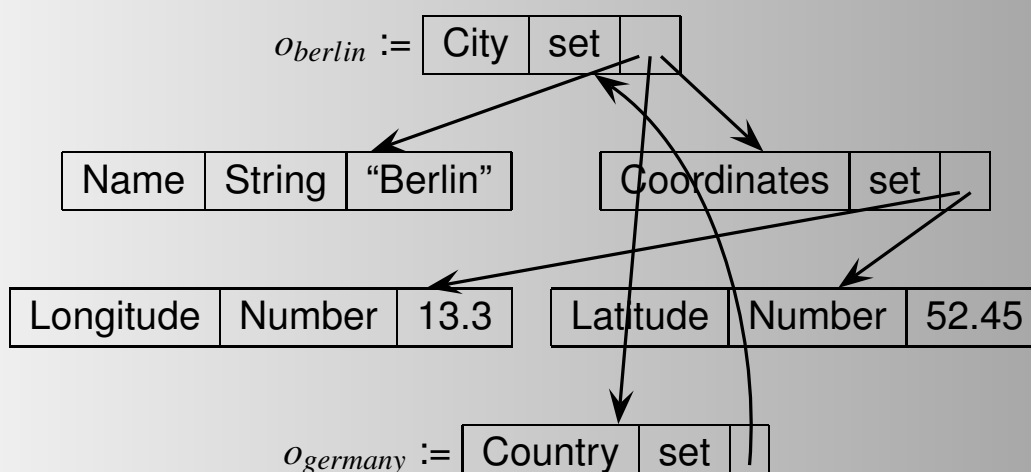
3

# Semistructured Data

- less fixed schema
- **self-describing – metadata are contained in the data**

OEM: Object Exchange Model (U Stanford, 1995)

- each object has a label, a type, and a value,
- complex values are represented as sets of references



# Document Data Model

- tree-like, nested
- flexible schema, certain structuring elements
- document = contents + markup
- markup-languages
  - logical markup: sectioning  $\Rightarrow$  tree structure
  - optical markup: fonts, colors
- well-known examples: HTML,  $\text{\LaTeX}$
- logical markup satisfies predefined constraints

XML

5

# HTML Example

```
<HTML>
  <HEAD><TITLE>Lecture: Computer Science I </TITLE></HEAD>
  <BODY>
    <H1>Computer Science I</H1>
    <UL><LI>Introduction </LI>
      <LI>Java</LI>
      <LI>Data Structures</LI>
    </UL>
    <P>Schedule:
    </P>
    <TABLE>
      <TR><TH>Date</TH><TH>Topic</TH></TR>
      <TR><TD>1.1.01</TD><TD><FONT COLOR="RED">Holiday</FONT></TD></TR>
      <TR><TD>8.1.01</TD><TD>Intro UNIX</TD></TR>
    </TABLE>
  </BODY>
</HTML>
```

XML

6

# XML: Requirements and Goals (~ 1996)

Processing and representation of semistructured data

Combination of

- database applications  
relational DB/SQL, OODB/OQL, OEM/OQL ...  
query languages, efficiency for large data sets
- document management  
SGML, HTML, transformation languages

⇒ flexible, expressive data model/language

Idea: data as contents + markup

SGML: expressive, flexible, complex

HTML: simple, concise, non-flexible

---

eXtensible Markup Language

“HTML with freely definable tags”

## XML Example: MONDIAL

<mondial>

```
<country car_code="B"  
  area="30510" capital="cty-Brussels"  
  memberships="org-eu org-nato ..">  
  <name>Belgium</name>  
  <population>  
    10170241  
  </population>  
  <city id="cty-Brussels"  
    country="B">  
    <name>Brussels</name>  
    <population year="95">  
      951580  
    </population>  
  </city>  
  :  
</country>
```

```
<organization id="org-eu" abbrev="EU"  
  headq="cty-Brussels">  
  <name>Europ. Union</name>  
  <abbrev>EU</abbrev>  
  <members type="member"  
    country="GR F E A D I B L ..."/>  
  <members type="applicant"  
    country="AL CZ ..."/>  
</organization>  
<organization id="org-nato" abbrev="NATO"  
  headq="cty-Brussels" ...>  
  :  
</organization>  
  :  
  :  
  :  
</mondial>
```

# XML: Requirements and Goals

Much more than “only” a markup language is required.

XML is more than only the Extensible Markup Language:

- The Markup Language serves only as string representation of XML data
  - compare (since early/mid 2000s): JSON (JavaScript Object Notation)
    - string representation of objects ... convertible into JavaScript objects
    - (actually, similar to the 1990s ODMG/CORBA “OIF – object interchange format” idea – the CORBA architecture itself was a middleware predecessor of the 2000s Web Services)
- The XML world provides much more ...

# XML: Requirements and Goals

Processing and representation of semistructured data

- flexible, expressive language: XML ✓
  - storage, queries
  - browsing, presentation
  - ⇒ data model
  - ⇒ efficient data structures and algorithms
  - ⇒ several kinds of languages to handle XML data
- “Internet-wide” data format
  - distributed, autonomous data sources
    - ⇒ standardization of interfaces
  - Electronic Data Interchange (cf. ODMG/CORBA) simple data transmission
    - ⇒ string representation (Unicode)

# XML as a Data Model/Data Structure

- relational data model is a data structure

compare: data structure “List”

- textual specification:  
A list is a sequence of elements ...
- abstract datatype:  
specifies a signature of operations and their algebraic specification (basic data manipulation language):  
create an empty list, append to a list, search in a list, ...
- implementations in class libraries
- representation as a character string:  
[1,4,9,16,25,36,49,63,81]

## The XML Data Model

“XML” is defined analogously:

- Idea: an abstract data model  
what information is contained in an XML document?  
W3C XML Information Set
- abstract datatype and implementations  
which operations?  
W3C Document Object Model (DOM)
- a character string representation  
how is XML data represented?  
data exchange format  
W3C XML

+ several languages to work with this data model

# The Abstract XML Data Model

- an XML *instance* is a tree (optionally also regarded as a nested structure)
- consisting of a lot of *nodes*
- of different *node types*.
- node type *document*: a distinguished root element.
- node type *element*: the tree structure consists of elements:
  - *element type* e.g. TABLE or country
  - *element contents*, among other things consisting of *subelements* (⇒ recursive structure),  
TABLE: TR-subelements; these again have TH and TD subelements
- node type *text*: most simple nodes in the element contents; text nodes are leafs.

XML

13

# The Abstract XML Data Model (cont'd)

- node type *attribute*: elements may have attributes
  - each *attribute node* is a (name, value)-pair:
  - it has an *attribute name* – color, car\_code
  - and an attribute value – “red”, “D”
  - different *attribute types*:  
(abbrev, “EU”), (area, “30510”): CDATA,  
(car\_code, “D”), (id, “cty-Brussels”), (id, “org-eu”): ID,  
(capital, “cty-Brussels”): IDREF,  
(members, “B D F”): IDREFS
- for text contents and attribute values: basic XML as a document format makes no distinction between strings, numbers, etc. – everything is just a string.

XML

14

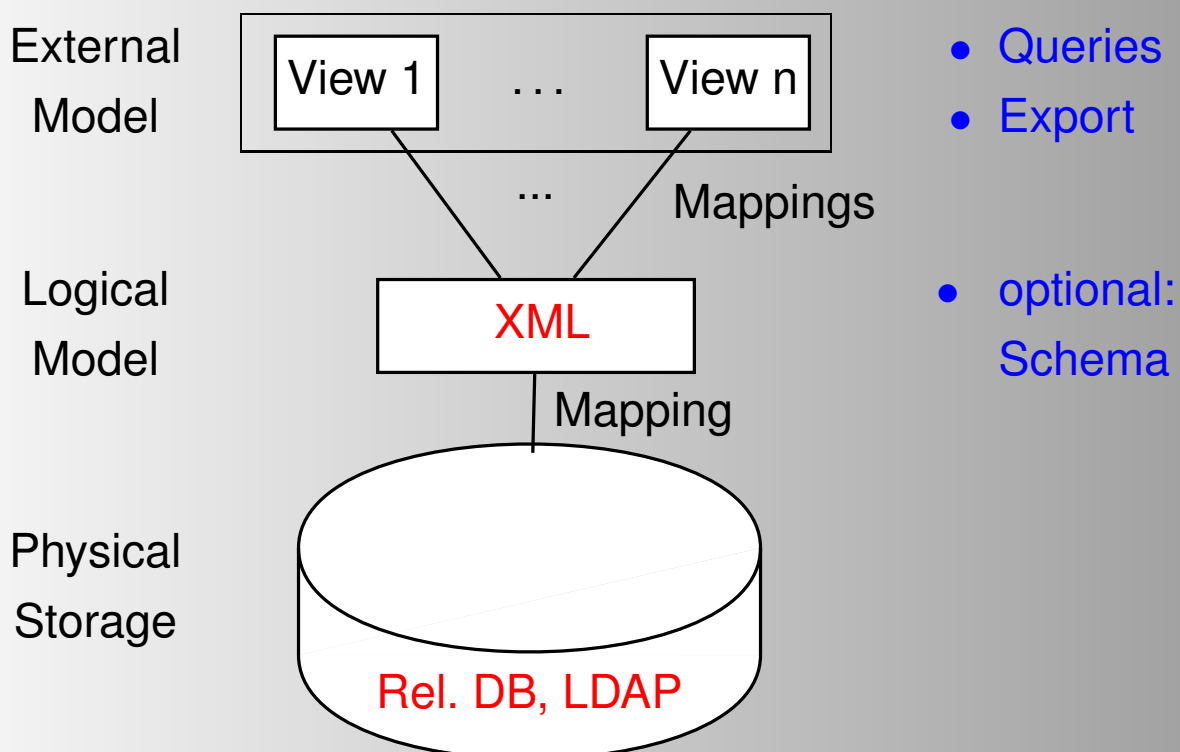
# The Document Object Model (DOM)

- defines a system of abstract datatypes:
  - **Document**
  - **Element**
  - **Attribute**
  - constructors
  - accessors, e.g. on elements:
    - return the type of the element (“country”, “population”)
    - traversing all child nodes (iterator)
    - access to attribute nodes (set + iterator)
- (reference) implementations in C++ (libxml) and Java
- different internal storage variants

XML

15

## 3-Level Architecture of DBS



XML

16



# The Character String Representation

- uses the same format as known from HTML
  - tags as parentheses `<country>...</country>`
  - text contents  
`<country><name>Germany</name></country>`
  - attributes `<country car_code="D">...</country>`
- can be kept in a file; is “human-readable”; can be edited;
- can be transmitted by simplest communication protocols;
- independent from
  - operating system
  - actual XML implementation
- **is only one representation of XML data**
- (from file or via data exchange) – must be parsed into a suitable (e.g. DOM) data structure before actual data processing

## XML Interfaces and Languages

Outlook to the next lessons:

“Family” of concepts around XML

- XML: definitions and details
- query languages
  - XPath: addressing, navigation
  - XQuery: queries (analogous to SQL)
- schema definition and description languages
  - DTDs: document-oriented XML applications
  - XML Schema: database-oriented XML
- transformation languages
  - XSL / XSLT stylesheets
- and some more