Chapter 4
XML (Extensible Markup Language)

Introduction

- SGML *very* expressive and flexible
  HTML very specialized.
- Summer 1996: John Bosak (Sun Microsystems) initiates the XML Working Group (SGML experts), cooperation with the W3C.
  Development of a subset of SGML that is simpler to implement and to understand
  `http://www.w3.org/XML/`: the homepage for XML at the W3C

⇒ XML is a "stripped-down version of SGML".

- for understanding XML, it is not necessary to understand everything about SGML ...

HTML

let's start the other way round: HTML ... well known, isn't it?

- tags: pairwise opening and closing: <TABLE> ... </TABLE>
- "empty" tags: without closing tag
, <HR>
- <P> is *in fact* not an empty tag (it should be closed at the end of the paragraph)!
- attributes: <TD colspan = "2"> ... </TD>
- empty tags with attributes:
  <IMG SRC="http://www.informatik.uni-goettingen.de/photo.jpg" ALIGN="LEFT">
- content of tag structures: <TD>123456</TD>
- nested tag structures: <TH><B>Name</B></TH>

  <A href="http:www.ifi.informatik.uni-goettingen.de">
  <B>Homepage of the IFI</B></A>

⇒ hierarchical structure

- Entities: ä = ä ß= ß

HTML

- browser must be able to interpret tags
  → semantics of each tag is fixed for all (?) browsers.
- fixed specifications how tags can be nested
  (described by a DTD (document type description))

  <body><H1><H2>
  <P> ... </P>
  <H2>
  <P> ... </P>
  <H1><H2>
  <P> ... </P>
  </body>
- analogously for tables and lists ...
- reality: people do in general not adhere to this structure
  - closing tags are omitted
  - structuring levels are omitted

  → parser has to be fault-tolerant and auto-completing

Knowledge of HTML for XML?

- intuitive idea – but only of the *ASCII representation*
- this is *not a data model*
- no query language
- only a very restricted viewpoint:
  HTML is a markup language for browsers
  (note: we don't "see" HTML in the browser, but only what the browser makes out of the HTML).

  Not any more.

Differences between XML and HTML?

- Goal: *not browsing*, but representation/storage of (semistructured) data (cf. SGML)
- SGML allows the definition of new tags according to the application semantics; each SGML application uses its own *semantic tags*.
  These are defined in a DTD (Document Type Description).
- HTML is *an* SGML application (cf. `<HTML> at the beginning of each document </HTML>`), that uses the DTD "HTML.dtd".
- In XML, (nearly) arbitrary tags can be defined and used:

```
<country> ... </country>
<city> ... </city>
<province> ... </province>
<name> ... </name>
```

- These *elements* represent objects of the application.

XML as a Meta-Language for Specialized Languages

- For each application, it can be chosen which "notions" are used as element names etc.:
  ⇒ document type definition (DTD)
- the set of allowed element names and their allowed nesting and attributes are defined in the DTD of the document (type).
- the DTD describes the *schema*
- XML is a *meta-language*, each DTD defines an own language
- for an application, either a new DTD can be defined, or an existing DTD can be used
  → standard-DTDs
- HTML has (as an SGML application) a DTD

Example: Mondial

```
<mondial>
    :
  <country code="D" capital="city-D-Berlin" memberships="EU NATO UN ...">
    <name>Germany</name>
    <encompassed continent="europe">100</encompassed>
    <population year="1995">83536115</population>
    <ethnicgroup name="German">95.1</ethnicgroup>
    <ethnicgroup name="Italians">0.7</ethnicgroup>
    <religion name="Roman Catholic">37</religion>
    <religion name="Protestant">45</religion>
    <language name="German">100</language>
    <border country="F" length="451"/>
    <border country="A" length="784"/>
    <border country="CZ" length="646"/>
      :
```


Example: Mondial (Forts.

```
       :
    <province id="prov-D-berlin" capital="city-D-berlin">
      <name>Berlin</name>
      <population year="1995">3472009</population>
      <city id="city-D-berlin">
        <name>Berlin</name> <population year="1995">3472009</population>
      </city>
    </province>
    <province id="prov-D-baden-wuerttemberg" capital="city-D-stuttgart">
      <population year="1995">10272069</population>
      <name>Baden Wuerttemberg</name>
      <city id="city-D-stuttgart">
        <name>Stuttgart</name> <population year="95">588482</population>
      </city>
      <city id="cty-D-mannheim"> ... </city>
      :
    </province>
    :
  </country>
  :
</mondial>
```

CHARACTERISTICS:

- hierarchical “data model”
- subelements, attributes
- references
- ordering? documents – yes, databases – no

Examples can be found at

`http://dbis.informatik.uni-goettingen.de/Mondial/#XML`

XML AS A DATA MODEL

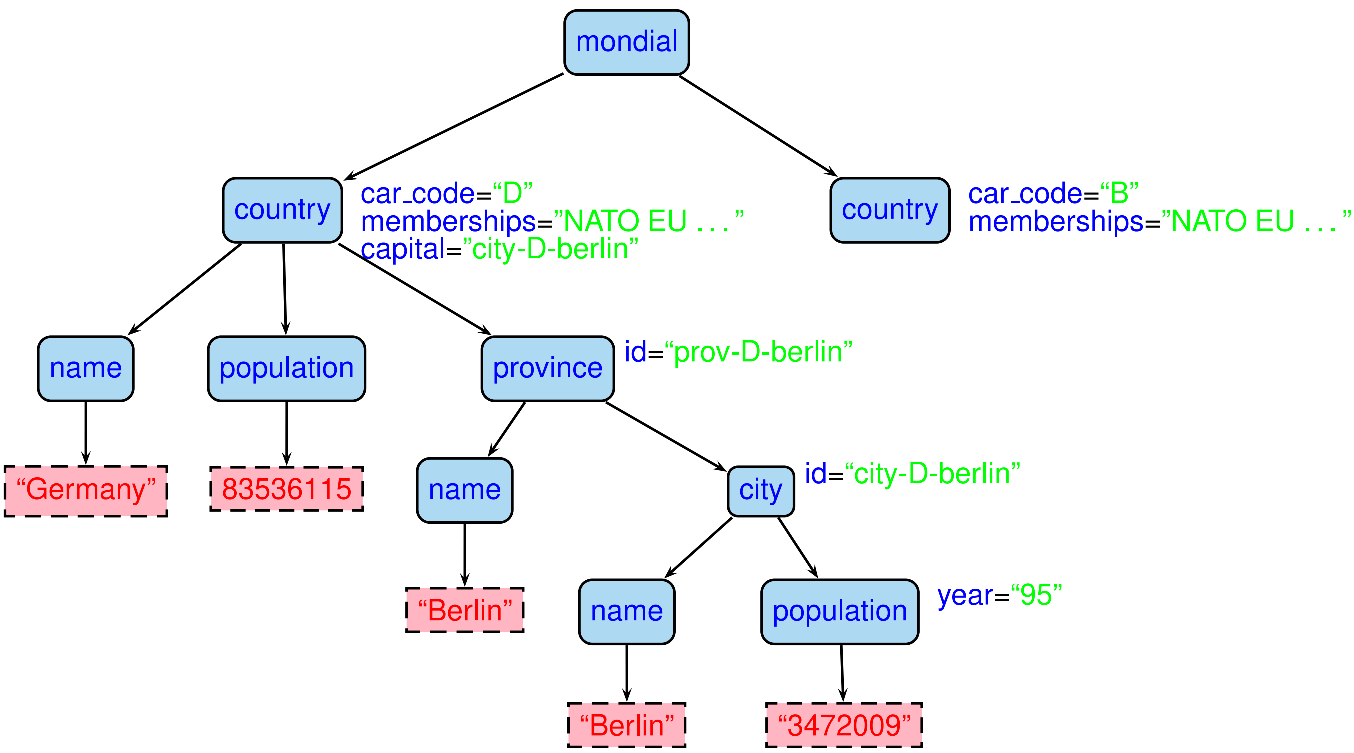
XML is much more than only the ASCII representation shown above as known from HTML (see also introductory talk)

- abstract data model (comparable to the relational DM)
- abstract datatype: DOM (Document Object Model) – see later
- many concepts around XML
  (XML is *not* a programming language!)
  - higher-level declarative query/manipulation language(s)
  - notions of “schema”

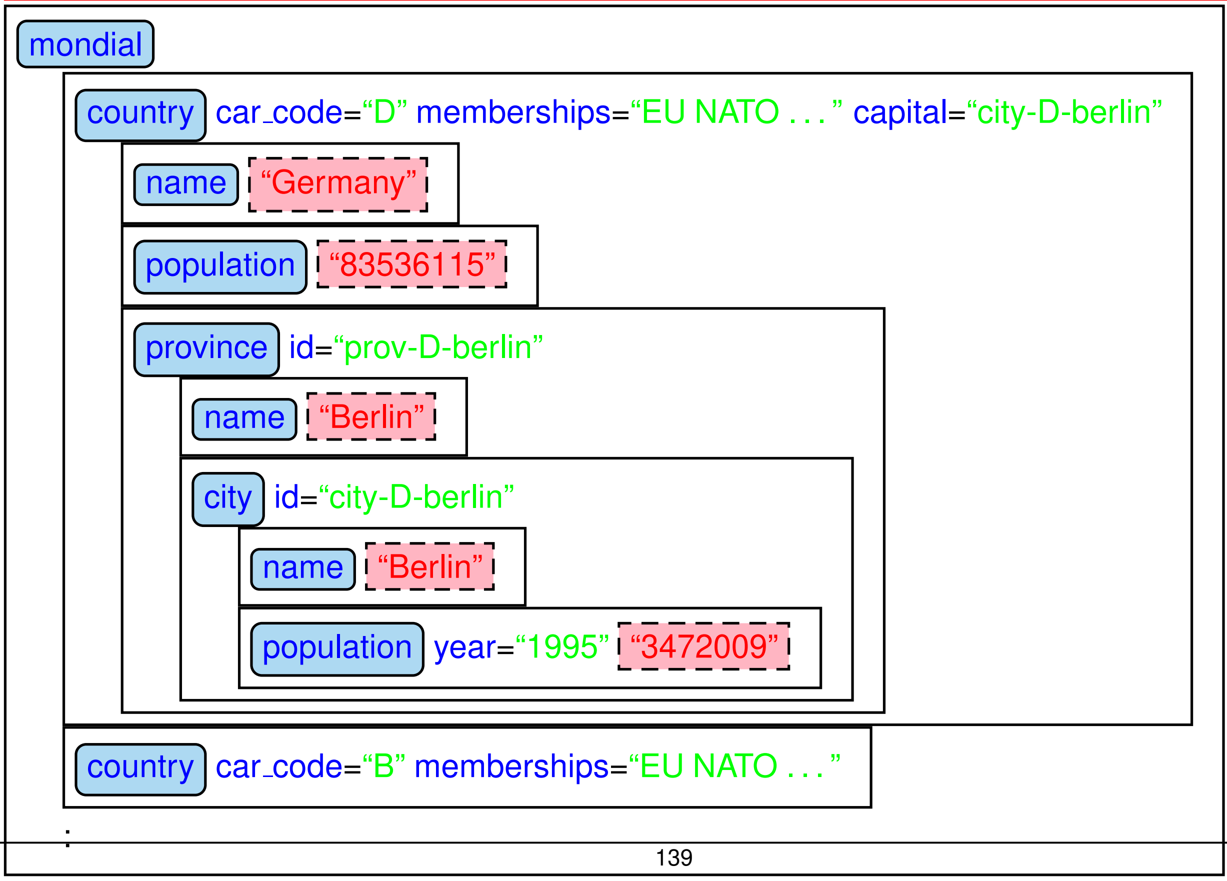
4.1 Structure of the Abstract XML Data Model (Overview)

- for each document there is a document node which “is” the document, and which contains information about the document (reference to DTD, doctype, encoding etc).
- the document itself consists of nested *elements* (tree structure),
- among these, exactly one *root element* that contains all other elements and which is the only child of the document node.
- elements have an element type (e.g. Mondial, Country, City)
- element content (if not empty) consists of text and/or *subelements*.
  These *child nodes* are ordered.
- elements may have *attributes*.
  Each attribute node has a name and a value (e.g. (car_code, “D”)).
  The attribute nodes are unordered.
- *empty elements* have no content, but can have attributes.
- a *node* in an XML document is a logical unit, i.e., an element, an attribute, or a text node.
- the allowed structure can be restricted by a schema definition.

Example: Mondial as a Tree



Example: Mondial as a Nested Structure



Observations

- there is a global order (preorder-depth-first-traversing) of all element- and text nodes, called *document order*.
- actual text is only present in the text-nodes
  Documents: if all text is concatenated in document order, a pure text version is obtained.
  Exercise: consider an HTML document.
- element nodes serve for structuring (but do not have a "value" for themselves)
- attribute nodes contain values whose semantics will be described in more detail later
  - attributes that describe the elements in more detail
    (e.g. td/@colspan or population/@year)
  - IDs and references to IDs
  - can be used for application-specific needs

ASCII: XML as a Representation Language

- elements are limited by
  - opening <Country> and
  - closing *tags* </Country>,
  - in-between, the *element content* is output recursively.
- Element content consists of text

  <Name> United Nations </Name>
- and *subelements*: <Country> <City> ... </City>

  <City> ... </City>

  </Country>
- *attributes* are given in the opening tag:

  <Country car_code="D"> . . . </Country>

  where attribute values are always given as strings, they do not have further structure. The difference between value- and reference attributes is not visible, but is only given by the DTD.
- *empty elements* have only attributes: <border country="F" length="451"/>

4.6 Summary and Outlook

XML: "basic version" consists of DTD and XML documents

- tree with additional cross references
- hierarchy of nested elements
- order of the subelements

  – documents: 1st, 2nd, . . . section etc.

  – databases: order in general not relevant
- attributes
- references via IDREF/IDREFS
  - documents: mainly cross references
  - databases: part of the data (relationships)
- XML model similar to the network data model:

  relationships are mapped into the structure of the data model
  - the basic explicit, stepwise navigation commands of the network data model have an equivalent for XML in the DOM-API (see later), but
  - XML also provides a declarative, high-level, set-oriented language.

Further Concepts of the XML World

Extensions:

- namespaces: use of different DTDs in a database
  (see Slide 207)
- APIs: DOM, SAX
- theoretical foundations
- query languages: XPath, XML-QL, Quilt, XQuery
- stylesheets/transformation languages: CSS, DSSSL, XSL
- better schema language: XML Schema
- XML with intra-document handling: XPointer, XLink

4.7 Recall

- XML as an abstract *data model*
  - cf. relational DM
  - XML now has become less abstract: creation of instances in the editor, validating, viewing ...
- a data model needs ... implementation? theory?
- ... first, something else: *abstract datatype, interface(s)*
  - constructors, modificators, selectors, predicates (cf. Info I)
- here: “two-level model”
  - as an ADT (programming interface): Document Object Model (DOM):
    detailed operations as usual in programming languages (Java, C++).
  - as a database model (end user interface; declarative):
    import (parser), *queries*, updates
- theory: formal specification of the semantics of the languages, other issues are the same as in classical DB theory (transactions etc.).

Chapter 5
Query Languages: XPath

- Network Data Model: no query language
- SQL – only for a flat data model, but a “nice” language
  (easy to learn, descriptive, relational algebra as foundation, clean theory, optimizations)
- OQL: SQL with object-orientation and path expressions
- Lorel (OEM): extension of OQL
- F-Logic: navigation in a graph by path expressions with additional conditions
  descriptive, complex.

XPath

1999: specification of the navigation formalism as *W3C XPath*.

- Base: UNIX directory notation

    in a UNIX directory tree: /home/dbis/Mondial/mondial.xml
    in an XML tree: /mondial/country/city/name

  Straightforward extension of the URL specification:
  /home/dbis/Mondial/mondial.xml#mondial/country/city/name
  [XPointer; later]
- W3C: XML Path Language (XPath), Version 1.0
  `http://www.w3.org/TR/xpath`
- W3C: XPath 2.0 and XQuery 1.0
  `http://www.w3.org/TR/xquery`
- Tools: see Web page (2004 course)
  - XML (XQuery) database system “eXist”
  - lightweight tool “saxonXQ” (XQuery)
  - embedded in XSLT stylesheet (using any XSLT processors)

XPath: Navigation, Simple Examples

XPath is based on the UNIX directory notation:

- /mondial/country
  addresses all country elements in Mondial,
  the result is a set of elements of the form

  <country code="..."> ... </country>

- /mondial/country/city
  addresses all city elements, that are direct subelements of country elements.
- /mondial/country//city
  adresses all city elements that are subelements of country elements.
- //city
  addresses all city elements in the current document.
- wildcards for element names:
  /mondial/country/*/capital
  addresses all capital elements that are grandchildren of country elements
  (different from /mondial/country//capital !)

... and now systematically:

XPath: Access Paths in XML Documents

- Navigation paths

  /step/step/.../step

  are composed by individual navigation steps,
- the result of each step is a set of nodes, that serve as input for the next step.
- each step consists of

  axis::*nodetest*[*condition*]

  - an axis (optional),
  - a nodetype-test,
  - a predicate (optional) that is evaluated for the current node.
- paths are combined by the "/"-operator
- additionally, there are function applications
- the result of each XPath expression is a *sequence* of nodes or literals.

XPATH: TESTS

In each step

path/*axis*::*nodetest*[*condition*]/*path*

condition is a predicate over XPath expressions.

- The expression selects only those nodes from the result of *path*/*axis*::*nodetest* that satisfy *condition*. *condition* contains XPath expressions that are evaluated relative to the current *context node* of the respective step.

  //country[@car_code="D"]

  returns the country element whose car_code attribute
  has the value "D"

- When comparing an element with something, the text() method is applied implicitly:

  //country[name = "Germany"] is equivalent to
  //country[name/text() = "Germany"]

- If the right hand side of the comparison is a number, the comparison is automatically evaluated on numbers:

  //country[population > 1000000]

XPATH: TESTS (CONT'D)

- boolean connectives "and" and "or" in *condition*:

  //country[population > 100000000 and @area > 5000000]
  //country[population > 100000000 or @area > 5000000]

- boolean "not" is a *function*:

  //country[not (population > 100000000)]

- XPath expressions in *condition* have existential semantics:
  The *truth value* associated with an XPath expression is *true*, if its result set is non-empty:

  //country[inflation]

  selects those countries that have a subelement of type inflation.

  ⇒ formal semantics: a path expression has

  - a semantics as a result set, and
  - a truth value!

XPATH: TESTS (CONT'D)

- XPath expressions in *condition* are not only “simple properties of an object”, but are path expressions that are evaluated wrt. the current context node:

  //city[population/@year=’95’]/name

- Such comparisons also have existential semantics:

  //country[.//city/name=’Cordoba’]/name

  returns the names of all countries, in which a city with name Cordoba is located.

  //country[not (.//city/name=’Cordoba’)]/name

  returns the names of those countries where no city with name Cordoba is located.