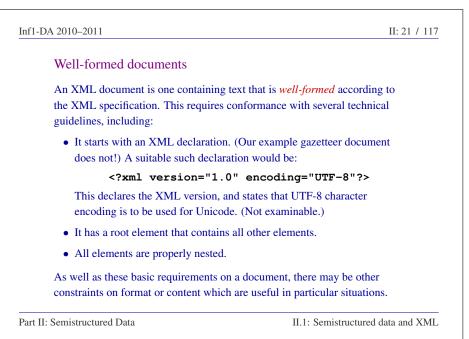
<capital>Ljubljana<region></region></capital>	tal>
<name>Gorenjska</name> <feature type="Lake">Ba</feature>	phinj
<feature mountain<="" th="" type="Mountain&lt;br&gt;&lt;Feature type="><th>n"&gt;Triglav</th></feature>	n">Triglav
 	•
data for other count:<br	ries here>
Part II: Semistructured Data	II.1: Semistructured data and XML

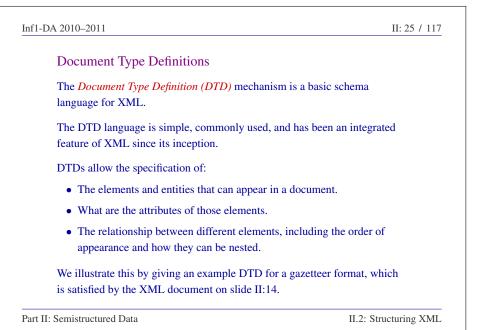
af1-DA 2010–2011	II: 20 / 117
Unicode	
An XML document is a text document	written in Unicode.
Unicode is a universal code for "text ch around 100,000 different characters.	aracters", currently supporting
The Unicode characters contain the star many, many other characters in use wor	
Each character has an assigned <i>code po</i> and 1,114,111 inclusive (hexadecimal 0	
The actual representation of Unicode te	ext in memory or "on the wire"
depends on a choice of <i>encoding</i> of Un	icode character sequences as byte
streams. The most common encoding is	s known as UTF-8; others include
UTF-16 and UTF-32.	
art II: Semistructured Data	II.1: Semistructured data and XML



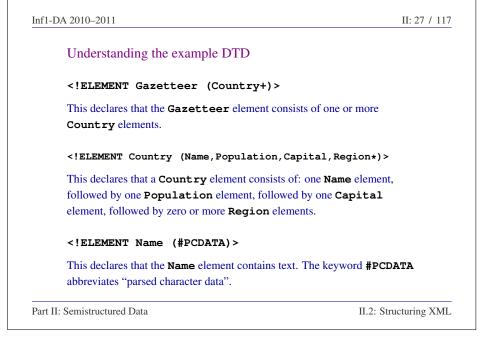
f1-DA 2010–2011	II: 22 / 117
Part II — Semistructured Data	
XML:	
II.1 Semistructured data, XPath and XML	
II.2 Structuring XML	
<b>II.3</b> Navigating XML using XPath	
Corpora:	
II.4 Introduction to corpora	
<b>II.5</b> Querying a corpus	
Related reading: §§4.1–4.3 of [XWT]	
§7.4.2 of [DMS]	
rt II: Semistructured Data	II.2: Structuring XMI

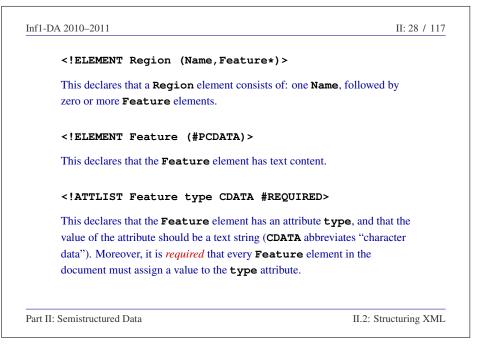
nf1-DA 2010–2011	II: 23 / 117
Structuring XML	
In a given XML application area, there is often an intende XML document should possess.	ed structure that an
For example, in the <b>Gazetteer</b> example, we expect the to respect the natural hierarchy:	various elements
• the Country elements are inside Gazetteer;	
<ul> <li>the Name (of the country), Population, Capital elements are inside Country;</li> </ul>	L and Region
• and the Name (of the region) and Feature elements Region.	s are inside
Moreover, the <b>Feature</b> elements assign a suitable value type.	to the attribute
Part II: Semistructured Data	II.2: Structuring XML

Inf1-DA 2010–2011	II: 24 / 117
Schema languages for XML	
In relational databases, a schema specifies the form	nat of a relation (table).
A <i>schema language</i> for XML is a language designer format of XML documents.	ed for specifying the
The use of a schema language has two main advant informal specification (cf. the informal and partial <b>Gazeteer</b> format on the previous slide):	
• It is precise and unambiguous	
• It is possible for a machine to check whether a satisfies a given schema specification ( <i>validati</i>	
If an XML document $X$ has the format specified b we say that $X$ is <i>valid</i> with respect to $S$ .	y a given schema $oldsymbol{S}$ then
Part II: Semistructured Data	II.2: Structuring XMI



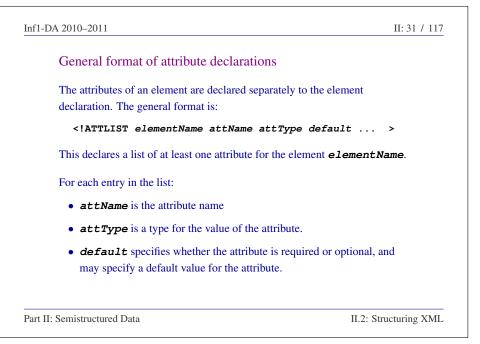
Inf1-DA 2010–2011	II: 26 / 117
Example DTD	
ELEMENT Gazetteer (Country+)	
ELEMENT Country (Name, Population, Ca</td <td><pre>apital,Region*)&gt;</pre></td>	<pre>apital,Region*)&gt;</pre>
ELEMENT Name (#PCDATA)	
ELEMENT Population (#PCDATA)	
ELEMENT Capital (#PCDATA)	
ELEMENT Region (Name,Feature*)	
ELEMENT Feature (#PCDATA)	
ATTLIST Feature type CDATA #REQUIRE</td <td>ID&gt;</td>	ID>
Part II: Semistructured Data	II.2: Structuring XML





Inf1-DA 2010–2011	II: 29 / 117
General format of element declarations	
An <i>element declaration</i> has the structure:	
ELEMENT elementName (contentType)	
There are four possible content types:	
1. <b>EMPTY</b> indicating that the element has no content, i.e. it is an <i>e element</i> as defined on slide II:16.	empty
<ol> <li>ANY indicating that any content is permitted. Nevertheless elements that appear within the element content n themselves be declared by corresponding element declarations.</li> </ol>	
<ul><li>3. <b>#PCDATA</b> indicating text content.</li><li>(In fact this is an instance of a more general <i>mixed content</i> form which we shall not consider further.)</li></ul>	nat,
Part II: Semistructured Data II.2:	Structuring XML

Inf1-DA 2010–2011	II: 30 / 117
4. A regular expression of element names.	
Regular expressions were introduced in Inf1 Computation and Logic.	
DTD's make use of the following format for regular expressions.	
• Any element name is a regular expression.	
(The element names are the <i>alphabet</i> for the regular expressions.)	
• exp1, exp2: first exp1 then exp2 in sequence.	
• <i>exp</i> *: zero or more occurrences of <i>exp</i> .	
• <i>exp</i> ?: zero or one occurrences of <i>exp</i> .	
• <b>exp+</b> : one or more occurrences of <b>exp</b> .	
• exp1   exp2: either exp1 or exp2.	
Part II: Semistructured Data II.2: Str	ructuring XMI

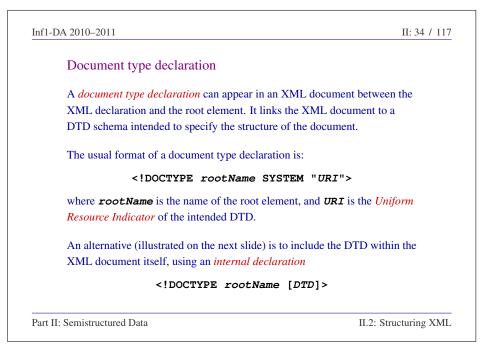


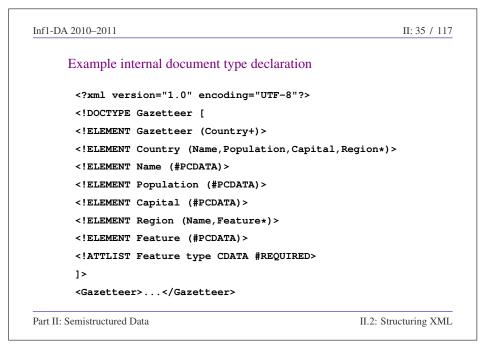
Inf1-DA 2010-2011 II: 32 / 117
We shall consider only the following attribute types:
String type: CDATA means that the attribute may have any text string as its value.
Enumerated type: (s<sub>1</sub> | s<sub>2</sub> | ... | s<sub>n</sub>) means that the attribute must take one of the strings s<sub>1</sub>, s<sub>2</sub>, ..., s<sub>n</sub> as its value.
And the following possibilities regarding default values:
Required: #REQUIRED means that the attribute must be explicitly assigned a value in every start tag for the element.
Optional: #IMPLIED means it is optional whether a value is assigned to the attribute or not.
Default: A fixed string can be specified as the default value for the attribute to take if no explicit value is given in the element's start tag.

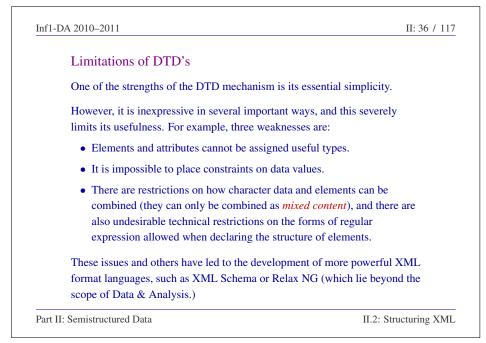
Part II: Semistructured Data

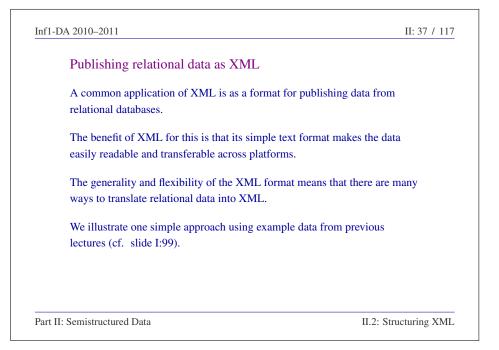
II.2: Structuring XML

Inf1-DA 2010–2011	II: 33 / 117
A variation on the example	
Consider replacing the attribute declaration in following declaration.	n the example DTD with the
ATTLIST Feature type (Mountain ]</td <td>Lake River) "Mountain"&gt;</td>	Lake River) "Mountain">
With this new (but not with the original) decl	aration:
<feature>Ben Nevis&lt;</feature>	/Feature>
would be a valid <b>Feature</b> element. The <b>ty</b> the default (and correct) default value <b>Mount</b>	
The element below is not valid with respect to valid for the original DTD)	o the new DTD (although it is
<feature type="Castle">Eilea</feature>	an Donan
because Castle is not one of the specified v	values for type.
Part II: Semistructured Data	II.2: Structuring XMI









Inf1-DA 2010-2011 II: 38 / 117 <?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE UniversityData [ <!ELEMENT UniversityData (Students, Courses, Takes)> <!ELEMENT Students (Student\*)> <!ELEMENT Student (mn, name, age, email) > <!ELEMENT Courses (C\*)> <!ELEMENT C (code, name, year)> <!ELEMENT Takes (T\*)> <!ELEMENT T (mn, name, mark) > <!ELEMENT mn (#PCDATA)> <!ELEMENT name (#PCDATA)> <!ELEMENT age (#PCDATA)> <!ELEMENT email (#PCDATA)> <!ELEMENT code (#PCDATA)> <!ELEMENT year (#PCDATA)> <!ELEMENT mark (#PCDATA)> 1> Part II: Semistructured Data II.2: Structuring XML

Inf1-DA 2010-2011 II: 39 / 117 <UniversityData> <Students> <Student> <mn>s0456782</mn> <name>John</name> <age>18</age> <email>john@inf</email> </Student> <Student> <mn>s0412375</mn> <name>Mary</name> <age>18</age> <email>mary@inf</email> </Student> <Student> <mn>s0378435</mn> <name>Helen</name> <age>20</age> <email>helen@phys</email> </Student> <Student> <mn>s0189034</mn> <name>Peter</name> <age>22</age> <email>peter@math</email> </Student> </Students> <Courses> <C><code>inf1</code><name>Informatics 1</name><year>1</year></C> <C><code>math1</code><name>Mathematics 1</name><year>1</year></C> </Courses> <Takes> <T><mn>s0412375</mn><code>inf1</code><mark>80</mark></T> <T><mn>s0378435</mn><code>math1</code><mark>70</mark></T> </Takes> </UniversityData> Part II: Semistructured Data II.2: Structuring XML

Inf1-DA 2010–2011	II: 40 / 117
Efficiency	
Relational database systems are optimised for storage efficiency.	
As we have seen, the XML version of relational data is extremely vert	oose.
Nevertheless, XML can still be stored efficiently using <i>data compressi</i> (which can be optimised for XML).	ion
Furthermore, once published XML data has been downloaded, it can be converted back to relational data so it can be stored efficiently in a loc database system.	
Converting XML to back to relational data has the benefit of enabling data to be queried ising relational database technology (i.e., SQL).	the
An interesting alternative is to apply newer technology for directly que XML.	erying
Part II: Semistructured Data II.2: St	ructuring XML