FAKULTÄT FÜR !NFORMATIK

Faculty of Informatics

Semi-structured Data

5 - XML Schema Definition (XSD)

Andreas Pieris and Wolfgang Fischl, Summer Term 2016

Outline

- XSDs at First Glance
- Validation
- A Reference to a Schema
- Schema Document Organization
- Simple Elements
- Attributes
- Restrictions on Content
- Complex Elements
- Order, Occurrence and Group Indicators
- Keys and References

XSD at First Glance

```
<person>
    <fullname> Andreas Pieris </fullname>
    <tel> 740072 </tel>
</person>
```

```
<!ELEMENT person (fullname, tel)>
<!ELEMENT fullname (#PCDATA)>
<!ELEMENT tel (#PCDATA)>
```

```
<?xml version="1.0"?>
<xsd:schema  xmlns:xsd=“http://www.w3.org/2001/XMLSchema”>
    <xsd:element  name=“person”>
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element  name=“fullname”   type=“xsd:string”/>
                <xsd:element  name=“tel”  type=“xsd:positiveInteger”/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

Validation

- Validating parsers - check both for well-formedness and validity
- Validating errors may be ignored (unlike well-formedness errors)
- Check for validity: xmllint - http://xmlsoft.org/
  - o Portable C library for Linux, Unix, MacOS, Windows, ...
  - o Command line call: **xmllint --valid <xml-file-name>**
  - o Check out http://www.dbai.tuwien.ac.at/education/ssd/current/uebung.html

A Reference to a Schema

- Referring to a DTD - Document Type Declaration

```
<?xml version="1.0"?>
<!DOCTYPE  person  SYSTEM “person.dtd”>
<person>
        <fullname>  Andreas Pieris </fullname>
        <tel> 740072 </tel>
</person>
```

A Reference to a Schema

- Referring to an XSD - hint in the instance document
  - o xsi:schemaLocation - list of namespaces, and the URIs of the schemas with which to validate the elements and attributes in those namespaces

```
<?xml version="1.0"?>
<person  xmlns="http://www.mysite.com"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.mysite.com   person.xsd">
    <fullname> Andreas Pieris </fullname>
    <tel> 740072 </tel>
</person>
```

A Reference to a Schema

- Referring to an XSD - hint in the instance document
    - xsi:noNamespaceSchemaLocation - a URL for the schema used to validate elements not in any namespace

```
<?xml version="1.0"?>
<person  xmlns:xsi=“http://www.w3.org/2001/XMLSchema-instance”
              xsi:noNamespaceSchemaLocation=“person.xsd”>
         <fullname> Andreas Pieris </fullname>
         <tel> 740072 </tel>
</person>
```

The xsd:schema Element

- Every schema document consists of a single root xsd:schema element
- The elements that make up an XML Schema must belong to the XML Schema namespace - usually associated with the prefix xsd: (or xs:)

```
<?xml version="1.0"?>
<xsd:schema  xmlns:xsd=“http://www.w3.org/2001/XMLSchema”>
        <xsd:element  name=“person”>
              <xsd:complexType>
                     <xsd:sequence>
                            <xsd:element  name=“fullname”  type=“xsd:string”/>
                            <xsd:element  name=“tel”  type=“xsd:positiveInteger”/>
                     </xsd:sequence>
              </xsd:complexType>
        </xsd:element>
</xsd:schema>
```

Global Elements

- Global Elements - appear at the top level of the schema (children of xsd:schema)
- May appear as the root of an instance document

```
<?xml version="1.0"?>
<xsd:schema  xmlns:xsd=“http://www.w3.org/2001/XMLSchema”>
        <xsd:element  name=“person”>
               <xsd:complexType>
                       <xsd:sequence>
                               <xsd:element  name=“fullname”  type=“xsd:string”/>
                               <xsd:element  name=“tel”  type=“xsd:positiveInteger”/>
                       </xsd:sequence>
               </xsd:complexType>
        </xsd:element>
</xsd:schema>
```

the only global element

Up to Now

- **XSDs at First Glance**
- **Validation**
- **A Reference to a Schema**
- **Schema Document Organization**
- Simple Elements
- Attributes
- Restrictions on Content
- Complex Elements
- Order, Occurrence and Group Indicators
- Keys and References

Simple Elements

- Contain only text - no other elements or attributes

- "Only text" is a bit misleading - several different data types
    - Build-in types (e.g., boolean, string, integer, etc.)

- Facets - we can add restrictions to a data type
    - Limit its content (e.g., min/max value)
    - Match a certain pattern (e.g., €ddd.dd)

Defining Simple Elements

```
<xsd:element name="element-name" type="element-type"/>
```

xsd:boolean, xsd:string, xsd:decimal, xsd:integer, xsd:date, xsd:time, etc.

`<fullname> Andreas Pieris </fullname>`	`<xsd:element name="fullname" type="xsd:string"/>`
`<tel> 740072 </tel>`	`<xsd:element name="tel" type="xsd:integer"/>`
`<dob> 1980-06-15 </dob>`	`<xsd:element name="dob" type="xsd:date"/>`
`<pass> yes </pass>`	`<xsd:element name="pass" type="xsd:boolean"/>`

Default and Fixed Values for Simple Elements

- Default value - assigned to the element when no other value is specified

```
<xsd:element name="element-name" type="element-type" default="default-value"/>
```

- Fixed value - assigned to the element, and no other value can be specified

```
<xsd:element name="element-name" type="element-type" fixed="fixed-value"/>
```

Attributes

- Simple elements cannot have attributes
- If an element has attributes, then it is of complex type (later)
- But the attribute itself is always of simple type

Defining Attributes

```
<xsd:attribute  name=“attribute-name”  type=“attribute-type”/>
```

xsd:boolean, xsd:string, xsd:decimal, xsd:integer, xsd:date, xsd:time, etc.

```
<fullname language=“EN”> Andreas Pieris </fullname>
```

```
<xsd:attribute  name=“language”  type=“xsd:string”/>
```

ATTENTION: We do not know yet how to define fullname (complex type)

Default and Fixed Values for Attributes

- Default value - assigned to the attribute when no other value is specified

```
<xsd:attribute  name="attribute-name"  type="attribute-type"  default="default-value"/>
```

- Fixed value - assigned to the attribute, and no other value can be specified

```
<xsd:attribute  name="attribute-name"  type="attribute-type"  fixed="fixed-value"/>
```

Optional and Required Attributes

```
<xsd:attribute  name=“attribute-name”  type=“attribute-type”  use=“optional”/>
```

OR

```
<xsd:attribute  name=“attribute-name”  type=“attribute-type”  use=“required”/>
```

ATTENTION: Attributes are optional by default

Restrictions on Content

- Several build-in datatypes
    - o Check out the textbook (XML in a Nutshell, Chapter 17)
- We can also add our own restrictions to elements and attributes
- These restrictions are called facets

Restrictions on Values

- minInclusive - greater than or equal
- maxInclusive - less than or equal
- minExclusive - greater than
- maxExclusive - less than

```
<xsd:element  name="age">
        <xsd:simpleType>
            <xsd:restriction  base="xsd:integer">
                <xsd:minExclusive  value="0"/>
                <xsd:maxInclusive  value="110"/>
            </xsd:restriction>
        </xsd:simpleType>
</xsd:element>
```

we create a new simple type by restricting the build-in type xsd:integer

Restrictions on Values

```
<xsd:element  name="age">
    <xsd:simpleType>
        <xsd:restriction  base="xsd:integer">
            <xsd:minExclusive   value="0"/>
            <xsd:maxInclusive   value="110"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
<xsd:element  name="duration">
    <xsd:simpleType>
        <xsd:restriction  base="xsd:integer">
            <xsd:minExclusive   value="0"/>
            <xsd:maxInclusive   value="110"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
```

Anonymous types

Restrictions on Values

```
<xsd:element  name=“age” type=“intervalType”/>
<xsd:element  name=“duration” type=“intervalType”/>

<xsd:simpleType  name=“intervalType”>
      <xsd:restriction  base=“xsd:integer”>
           <xsd:minExclusive  value=“0”/>
           <xsd:maxInclusive  value=“110”/>
      </xsd:restriction>
</xsd:simpleType>
```

Named type

ATTENTION: Named types are recommended - reusability

Restrictions on a Set of Values

- enumeration - limit the content to a set of acceptable values

```
<xsd:element  name="color" type="rgbType"/>

<xsd:simpleType  name="rgbType">
    <xsd:restriction  base="xsd:string">
        <xsd:enumeration  value="Red"/>
        <xsd:enumeration  value="Green"/>
        <xsd:enumeration  value="Blue"/>
    </xsd:restriction>
</xsd:simpleType>
```

Restrictions on a Series of Values

- pattern - limit the content to a certain sequence of characters

```
<xsd:element name=“pin” type=“pinType”/>

<xsd:simpleType name=“pinType”>
        <xsd:restriction base=“xsd:integer”>
              <xsd:pattern value=“[0-9][0-9][0-9][0-9]”/>
        </xsd:restriction>
</xsd:simpleType>
```

Restrictions on a Series of Values

- “[A-Z][A-Z][A-Z]” - triples of uppercase letters from A to Z
- “[a-zA-Z][a-zA-Z][a-zA-Z]” - triples of lowercase/uppercase letters from A to Z
- “[abcd]” - one of the letters a, b, c or d
- “([a-z])*” - zero or more occurrences of lowercase letters from a to z
- “([a-z][A-Z])+” - one or more occurrences of pairs of letters (e.g., sToP, mOrE)
- “male | female” - either male or female
- “[a-zA-Z0-9]{5}” - exactly 5 characters of letters or numbers from 0 to 9

Restrictions on Whitespace Characters

- whiteSpace - specifies how whitespace characters (line feeds, tabs, spaces, and carriage returns) are handled

```
<xsd:element name="definition" type="defType"/>

<xsd:simpleType name="defType">
    <xsd:restriction base="xsd:string">
        <xsd:whiteSpace value="preserve"/>
    </xsd:restriction>
</xsd:simpleType>
```

preserve	-	keep whitespace characters
replace	-	replace whitespace characters with space
collapse	-	remove all whitespace characters

Restrictions on Length

- length, minLength, maxLength - limit the length of a value in an element

```
<xsd:element  name=“password” type=“pswType”/>

<xsd:simpleType  name=“pswType”>
      <xsd:restriction  base=“xsd:string”>
            <xsd:minLength  value=“4”/>
            <xsd:maxLength  value=“8”/>
      </xsd:restriction>
</xsd:simpleType>
```

Restrictions for Datatypes - Sum Up

Constraint	Description
minInclusive	Greater or equal than
maxInclusive	Less or equal than
minExclusive	Greater than
maxExclusive	Less than
enumeration	Set of acceptable values
pattern	Certain sequence of characters
whiteSpace	Specifies how whitespace characters are handled
length	Exact number of characters
minLength	Minimum number of characters
maxLength	Maximum number of characters

Up to Now

- **XSDs at First Glance**
- **Validation**
- **A Reference to a Schema**
- **Schema Document Organization**
- **Simple Elements**
- **Attributes**
- **Restrictions on Content**
- Complex Elements
- Order, Occurrence and Group Indicators
- Keys and References

Complex Elements

- Contain other elements and/or attributes

- Four kinds of complex elements
  - Empty elements
  - Elements that contain only other elements (elements only)
  - Elements that contain only text (text only)
  - Elements that contain both elements and text (mixed)

ATTENTION: Each of these elements may contain attributes as well

Defining Complex Empty Elements

```
<person id="E832740"/>
```

```
<xsd:element name="person" type="personType"/>

<xsd:complexType name="personType">
        <xsd:attribute name="id" type="xsd:ID"/>
</xsd:complexType>
```

we create a new complex type

ATTENTION: Complex types can be anonymous or named (like simple types)

Defining Complex “Element-only” Elements

```
<person>
        <firstname> Andreas </firstname>
        <lastname> Pieris </lastname>
</person>
```

```
<xsd:element  name=“person”  type=“personType”/>

<xsd:complexType  name=“personType”>
        <xsd:sequence>
                <xsd:element  name=“firstname”  type=“xsd:string”/>
                <xsd:element  name=“lastname”  type=“xsd:string”/>
        </xsd:sequence>
</xsd:complexType>
```

Defining Complex “Text-only” Elements

- Text and attributes - we add a simpleContent element around the content

```
<xsd:element  name=“element-name”  type=“newType”/>

<xsd:complexType  name=“newType”>
        <xsd:simpleContent>
               <xsd:extension  base=“type”>
                      …
               </xsd:extension>
        </xsd:simpleContent>
</xsd:complexType>
```

Defining Complex "Text-only" Elements

```
<person id="E832740"> Andreas Pieris </person>
```

```
<xsd:element name="person" type="personType"/>

<xsd:complexType name="personType">
    <xsd:simpleContent>
        <xsd:extension base="xsd:string">
            <xsd:attribute name="id" type="xsd:ID"/>
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>
```

we create a new complex type which:
- o allows only for simple content, and
- o extends xsd:string by adding an attribute

Defining Complex “Mixed-content” Elements

```
<definition>
        The term <term> Semi-structured Data </term>
        refers to a form of structured data that does not
        conform with the formal structure of relational data
</definition>
```

mixed content

```
<xsd:element name=“definition” type=“definitionType”/>

<xsd:complexType name=“definitionType” mixed=“true”>
        <xsd:sequence>
                <xsd:element name=“term” type=“xsd:string”/>
        </xsd:sequence>
</xsd:complexType>
```

specifies the order in which the child elements must appear

Indicators

- Order indicators - to define the order of the elements
- Occurrence indicators - to define how often an element can occur
- Group indicators - to define related sets of elements
    - Check out the textbook (XML in a Nutshell, Chapter 17)

Order Indicators

- all - the child elements can appear in any order, while each child element can appear only once

```
<xsd:element  name=“person”  type=“personType”/>

<xsd:complexType  name=“personType”>
     <xsd:all>
          <xsd:element  name=“firstname”  type=“xsd:string”/>
          <xsd:element  name=“lastname”  type=“xsd:string”/>
     </xsd:all>
</xsd:complexType>
```

```
<person>
    <firstname> Andreas </firstname>
    <lastname> Pieris </lastname>
</person>          ✓
```

```
<person>
    <lastname> Pieris </lastname>
    <firstname> Andreas </firstname>
</person>          ✓
```

Order Indicators

- all - the child elements can appear in any order, while each child element can appear only once

```
<xsd:element name="person" type="personType"/>

<xsd:complexType name="personType">
    <xsd:all>
        <xsd:element name="firstname" type="xsd:string"/>
        <xsd:element name="lastname" type="xsd:string"/>
    </xsd:all>
</xsd:complexType>
```

```
<person>
    <firstname> Andreas </firstname>
    <firstname> Pieris </firstname>
</person>        ✘
```

```
<person>
    <firstname> Andreas </firstname>
    <lastname> Pieris </lastname>
    <lastname> Pieris </lastname>
</person>        ✘
```

Order Indicators

- choice - exactly one child element, is interpreted as XOR

```
<xsd:element  name="person"  type="personType"/>

<xsd:complexType  name="personType">
      <xsd:choice>
            <xsd:element  name="firstname"  type="xsd:string"/>
            <xsd:element  name="lastname"  type="xsd:string"/>
      </xsd:choice>
</xsd:complexType>
```

```
<person>
   <firstname> Andreas </firstname>
</person>
```

✓

```
<person>
   <lastname> Pieris </lastname>
</person>
```

✓

Order Indicators

- choice - exactly one child element, is interpreted as XOR

```
<xsd:element  name="person"  type="personType"/>

<xsd:complexType  name="personType">
      <xsd:choice>
            <xsd:element  name="firstname"  type="xsd:string"/>
            <xsd:element  name="lastname"  type="xsd:string"/>
      </xsd:choice>
</xsd:complexType>
```

```
<person>
   <firstname> Andreas </firstname>
   <lastname> Pieris </lastname>
</person>          ✗
```

```
<person>
   <lastname> Pieris </lastname>
   <firstname> Andreas </firstname>
</person>          ✗
```

Order Indicators

- sequence - the child element must appear in a specific order

```
<xsd:element name="person" type="personType"/>

<xsd:complexType name="personType">
        <xsd:sequence>
                <xsd:element name="firstname" type="xsd:string"/>
                <xsd:element name="lastname" type="xsd:string"/>
        </xsd:sequence>
</xsd:complexType>
```

… we have already seen sequence several times

Occurrence Indicators

- minOccurs - the minimum number of times an element can occur
- maxOccurs - the maximum number of times an element can occur

<xsd:element name="element-name" type="element-type"

minOccurs="N_1" maxOccurs="N_2"/>

ATTENTION: maxOccurs="unbounded" - unbounded number of times

Keys and References

- Let’s go back to DTDs for a moment

```
<!ATTLIST  employee  emp_id  ID  #REQUIRED>
<!ATTLIST  project  proj_id  ID  #REQUIRED>
<!ATTLIST  manager  mgr_id  IDREF  #REQUIRED>
<!ELEMENT  employee  (#PCDATA)>
<!ELEMENT  project  (#PCDATA)>
<!ELEMENT  manager  (#PCDATA)>
```

```
<employee emp_id=“e1”> E </employee>
<project proj_id=“p1”> P </project>
<manager mgr_id=“e1”> E </manager>
```

managers are employees

Keys and References

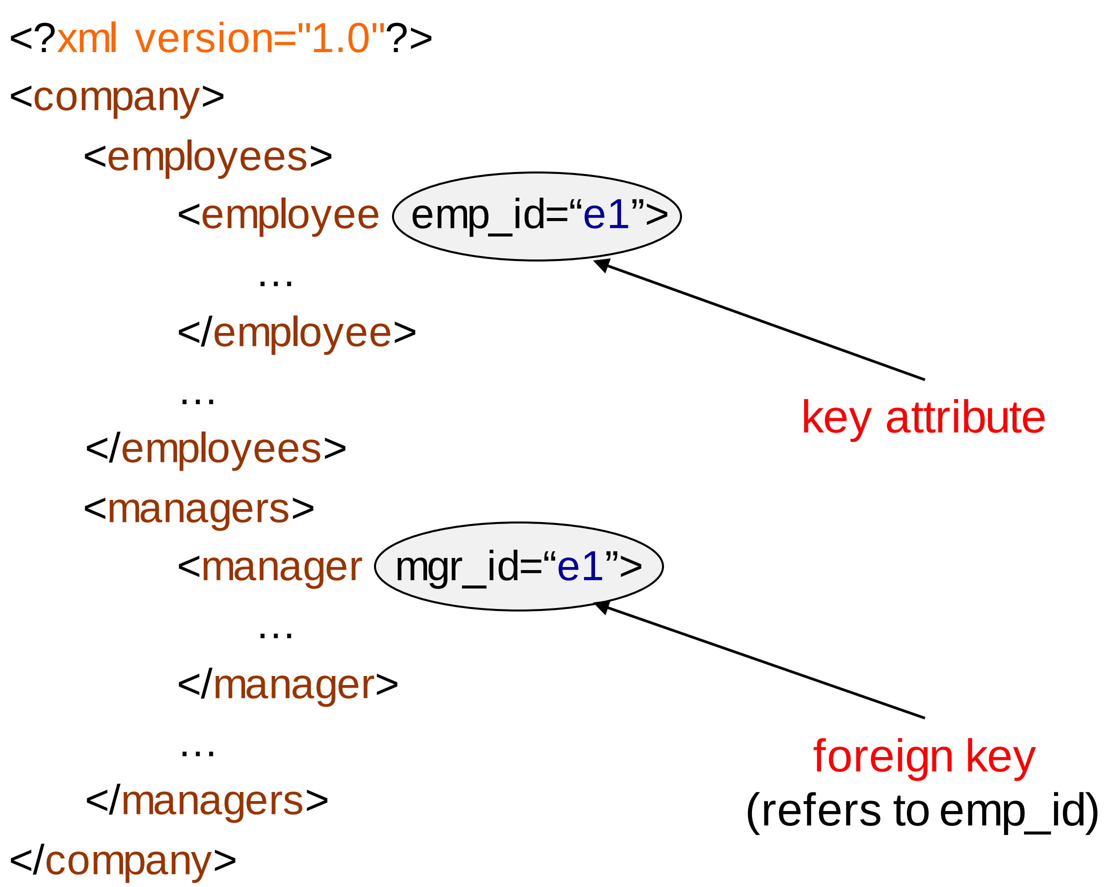
- Let’s go back to DTDs for a moment

```
<!ATTLIST employee emp_id ID #REQUIRED>
<!ATTLIST project proj_id ID #REQUIRED>
<!ATTLIST manager mgr_id IDREF #REQUIRED>
<!ELEMENT employee (#PCDATA)>
<!ELEMENT project (#PCDATA)>
<!ELEMENT manager (#PCDATA)>
```

```
<employee emp_id=“e1”> E </employee>
<project proj_id=“p1”> P </project>
<manager mgr_id=“p1”> E </manager>
```

valid, although conceptually wrong
(manager is a project)

Keys and References



```
<?xml version="1.0"?>
<company>
    <employees>
        <employee emp_id=“e1”>
            …
        </employee>
        …
    </employees>
    <managers>
        <manager mgr_id=“e1”>
            …
        </manager>
        …
    </managers>
</company>
```

Keys and References

```
<xsd:element  name=“company”  type=“companyType”>
      <xsd:key  name=“empKey”>
            <xsd:selector  xpath=“employees/employee”/>
            <xsd:field  xpath=“@emp_id”/>
      </xsd:key>

      <xsd:keyref  name=“empRef”  refer=“empKey”>
            <xsd:selector  xpath=“managers/manager”/>
            <xsd:field  xpath=“@mgr_id”/>
      </xsd:keyref>
</xsd:element>

<xsd:complexType  name=“companyType”>
      …
</xsd:complexType>
```

select emp_id

select mgr_id

XPath expressions
(week 7)

Sum Up

- XSDs at First Glance
- Validation
- A Reference to a Schema
- Schema Document Organization
- Simple Elements
- Attributes
- Restrictions on Content
- Complex Elements
- Order, Occurrence and Group Indicators
- Keys and References