Grammars for XML Documents

XML Schema, Part 2

Lecture "XML in Communication Systems"
Chapter 5

Dr.-Ing. Jesper Zedlitz
Research Group for Communication Systems
Dept. of Computer Science
Christian-Albrechts-University in Kiel

Overview

1. Brief Note on Terminology
2. Attributes
3. Content Models
4. Element Declaration
5. Uniqueness and Keys
6. Substitution
7. Abstraction

Chapter 5.1

Brief Note on Terminology

Brief Note on Terminology

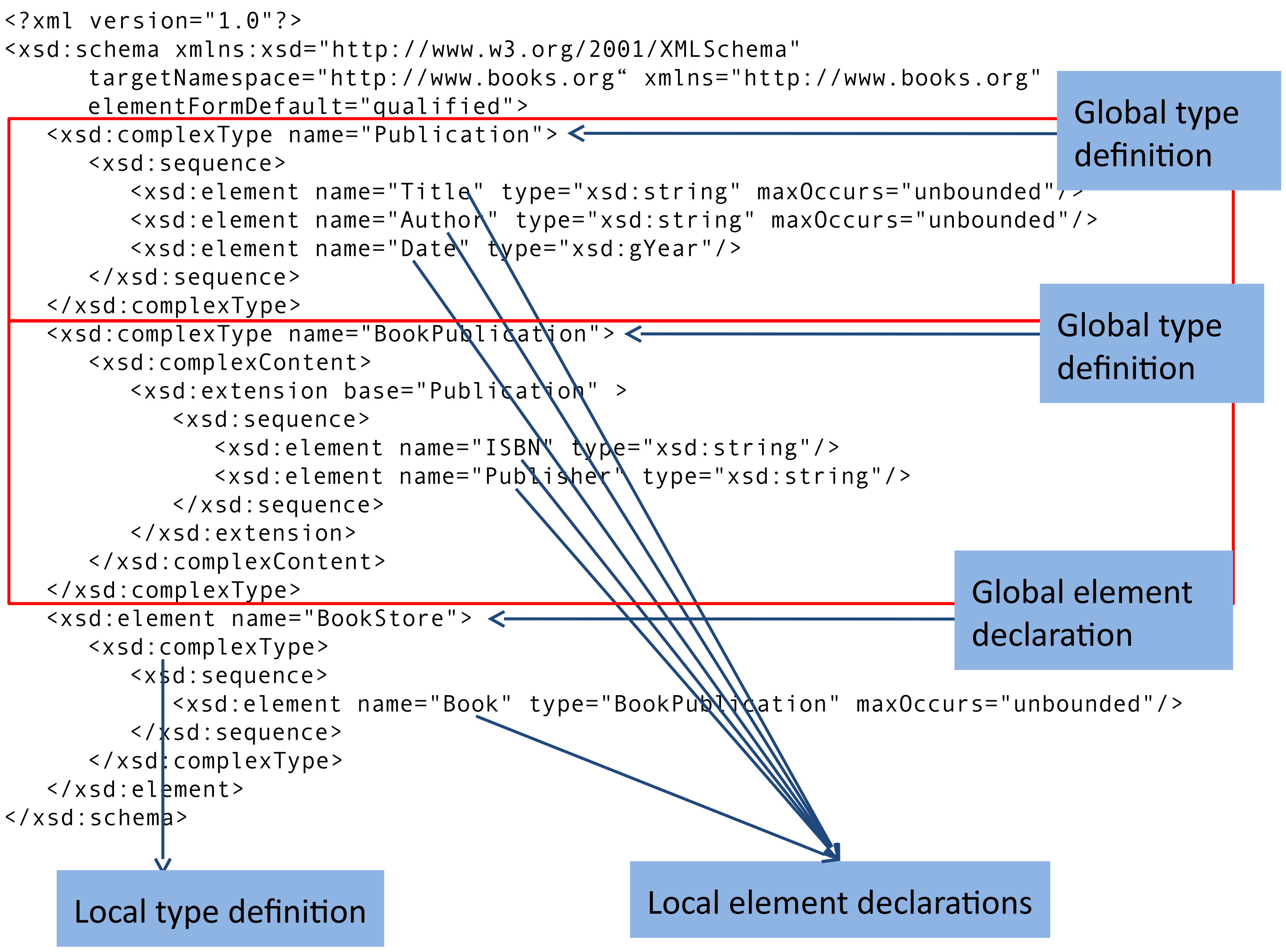
- Declaration vs Definition
  - In a schema:
    - You *declare* elements and attributes. Schema components that are *declared* are those that have a representation in an XML instance document.
    - You *define* components that are used just within the schema document(s). Schema components that are *defined* are those that have no representation in an XML instance document.

Declarations:	**Definitions:**
- element declarations	- type (simple, complex) definitions
- attribute declarations	- attribute group definitions
	- model group definitions

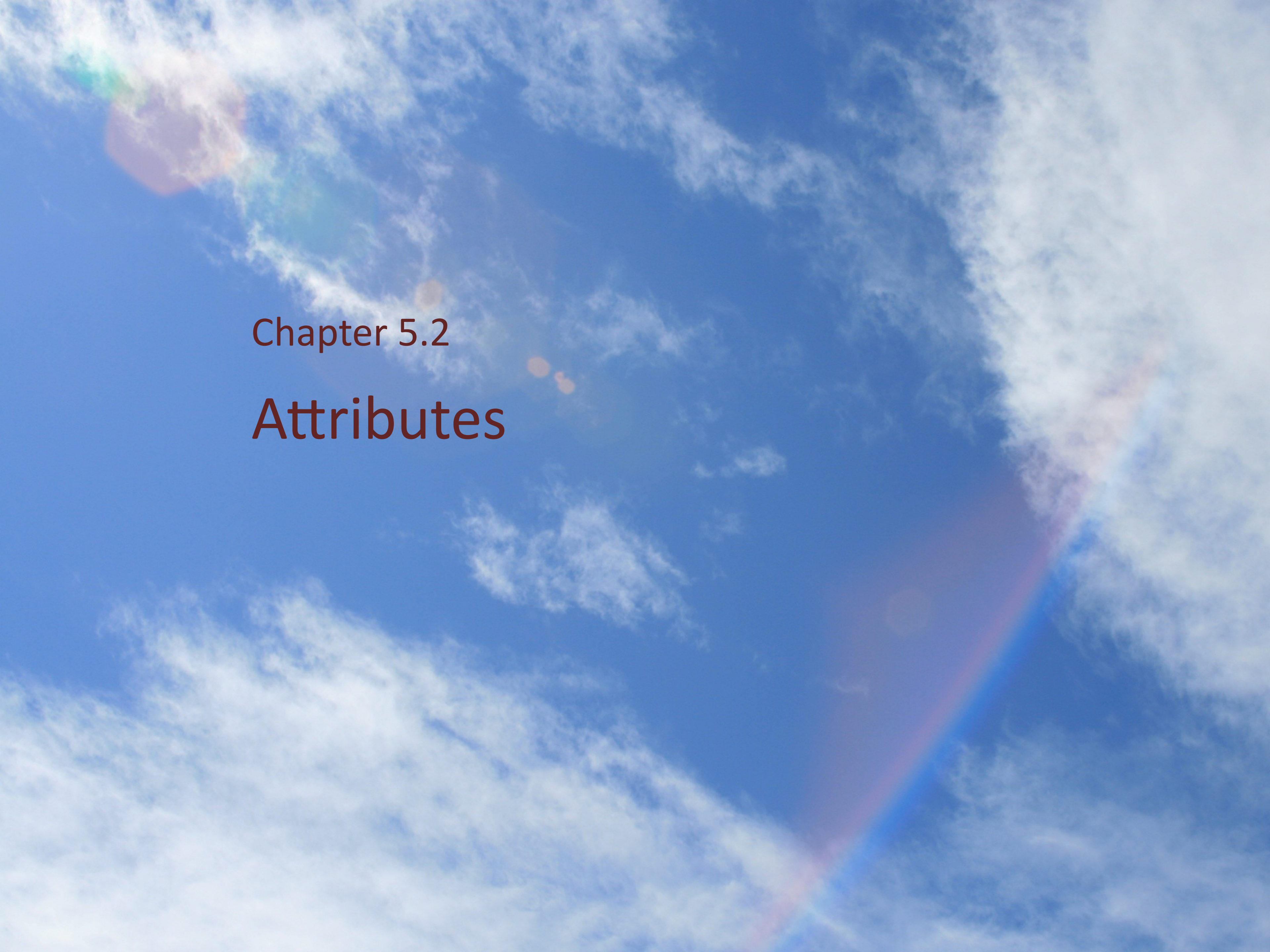
© N. Luttenberger

Brief Note on Terminology

- Global versus Local
  - Global element declarations, global type definitions:
    - These are element declarations/type definitions that are immediate children of <schema>
  - Local element declarations, local type definitions:
    - These are element declarations/type definitions that are nested within other elements/types.
  - Only global elements/types can be referenced (i.e., reused). Local elements/types are effectively invisible to the rest of the schema (and to other schemas).



```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      targetNamespace="http://www.books.org“ xmlns="http://www.books.org"
      elementFormDefault="qualified">
   <xsd:complexType name="Publication">
      <xsd:sequence>
         <xsd:element name="Title" type="xsd:string" maxOccurs="unbounded"/>
         <xsd:element name="Author" type="xsd:string" maxOccurs="unbounded"/>
         <xsd:element name="Date" type="xsd:gYear"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="BookPublication">
      <xsd:complexContent>
         <xsd:extension base="Publication" >
            <xsd:sequence>
               <xsd:element name="ISBN" type="xsd:string"/>
               <xsd:element name="Publisher" type="xsd:string"/>
            </xsd:sequence>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
   <xsd:element name="BookStore">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="Book" type="BookPublication" maxOccurs="unbounded"/>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
</xsd:schema>
```



Chapter 5.2

Attributes

Declaring Attributes

- Basics

1 `<xsd:attribute name/ref="`*name*`" type="`*simple-type*`" use="`*how-its-used*`" default/fixed="`*value*`"/>`

```
xsd:string
xsd:integer
xsd:boolean
. . .
```
(Attributes cannot have child elements.)

```
required
optional
prohibited
```

The `use` attribute value must be `optional` if you use `default` or `fixed`.

More attributes: next slide!

2 attribute value has embedded derived type

```
<xsd:attribute name="name" use="how-its-used" default/fixed="value">
    <xsd:simpleType>
        <xsd:restriction base="simple-type">
            …
```

Attribute Declaration Attributes

name	declares the name of the attribute in instance documents.
ref	refers a global attribute declaration.
type	declares the datatype of the attribute in instance documents.
default	gives the attribute a default value.
fixed	gives the attribute a fixed (constant) value.
use	specifies whether the attribute is `optional`, `required`, or `prohibited`. If a default or fixed value is specified, then **use** must have the value `optional`. The default value of **use** is `optional`.
id	associates a unique identifier to the attribute. This is purely internal to the schema. The type of this value must be `xsd:ID`.
form	overrides the `attributeFormDefault` schema attribute. **form** attribute values are either `"qualified"` or `"unqualified"`.

Declaring Attributes

- Embedded derived attribute type

```
<xsd:attribute name="Category" use="required">
   <xsd:simpleType>
      <xsd:restriction base="xsd:string">
         <xsd:enumeration value="autobiography"/>
         <xsd:enumeration value="non-fiction"/>
         <xsd:enumeration value="fiction"/>
      </xsd:restriction>
   </xsd:simpleType>
</xsd:attribute>
```

  – Instance documents are required to have the `Category` attribute (as indicated by use="required").

  – The value of `Category` must be either autobiography, non-fiction, or fiction (as specified by the enumeration facets).

Declaring Attributes

- No datatype specified
  - You can declare an attribute without specifying a datatype, e.g.:

    <attribute name="shape"/>
  - This attribute is unconstrained with respect to the type of value it can have.
  - Creating unconstrained attributes is a very useful technique:
    - Consider creating a complexType containing unconstrained attributes, and then create other complexTypes which derive by restriction and which constrain the shape attribute to a specific datatype. That's nice!

© N. Luttenberger

Attributes

- `use="prohibited"` example

"master"

```
<xsd:complexType name="shape">
   <xsd:attribute name="length" type="xsd:nonNegativeInteger"/>
   <xsd:attribute name="height" type="xsd:nonNegativeInteger"/>
   <xsd:attribute name="width" type="xsd:nonNegativeInteger"/>
   <xsd:attribute name="radius" type="xsd:nonNegativeInteger"/>
   <xsd:attribute name="diameter" type="xsd:nonNegativeInteger"/>
</xsd:complexType>
```

"subtype"

```
<xsd:complexType name="box">
   <xsd:complexContent>
      <xsd:restriction base="shape">
         <xsd:attribute name="length" type="xsd:nonNegativeInteger"/>
         <xsd:attribute name="height" type="xsd:nonNegativeInteger"/>
         <xsd:attribute name="width" type="xsd:nonNegativeInteger"/>
         <xsd:attribute name="radius" type="xsd:nonNegativeInteger" use="prohibited"/>
         <xsd:attribute name="diameter" type="xsd:nonNegativeInteger" use="prohibited"/>
      </xsd:restriction>
   </xsd:complexContent>
</xsd:complexType>
```

Local and Global Attributes

- Attributes may be declared locally or globally
  - local attribute declarations:
    - Inline an attribute declaration within a complexType.
  - global attribute declarations
    - attribute declarations are immediate children of <schema>

© N. Luttenberger

Local and Global Attributes

```
<xsd:schema … >
   <xsd:element name="Book">
      <xsd:complexType>
         <xsd:sequence>
            …
         </xsd:sequence>
         <xsd:attribute ref="Category" use="required"/>
         …
      </xsd:complexType>
   </xsd:element>

   <xsd:attribute name="Category">
      <xsd:simpleType>
         <xsd:restriction base="xsd:string">
            <xsd:enumeration value="autobiography"/>
            <xsd:enumeration value="fiction"/>
            <xsd:enumeration value="non-fiction"/>
         </xsd:restriction>
      </xsd:simpleType>
   </xsd:attribute>
```

Local attribute declaration.

Notice the "ref"

Global attribute declaration.

Must NOT have a "use": "use" only makes sense in the context of an element

Declaring Attributes Locally

```
<xsd:element name="Book" maxOccurs="unbounded">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Title" type="xsd:string"/>
      <xsd:element name="Author" type="xsd:string" maxOccurs="unbounded"/>
      <xsd:element name="Date" type="xsd:string"/>
      <xsd:element name="ISBN" type="xsd:string"/>
      <xsd:element name="Publisher" type="xsd:string"/>
    </xsd:sequence>
    <xsd:attribute name="Category" use="required">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="autobiography"/>
          <xsd:enumeration value="non-fiction"/>
          <xsd:enumeration value="fiction"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="InStock" type="xsd:boolean" default="false"/>
    <xsd:attribute name="Reviewer" type="xsd:string" default=" "/>
  </xsd:complexType>
</xsd:element>
```

Local (inlined) attributes

Declaring Attributes Locally

```
<xsd:element name="Book">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Title" type="xsd:string"/>
      <xsd:element name="Author" type="xsd:string" maxOccurs="unbounded"/>
      <xsd:element name="Date" type="xsd:string"/>
      <xsd:element name="ISBN" type="xsd:string"/>
      <xsd:element name="Publisher" type="xsd:string"/>
    </xsd:sequence>
    <xsd:attribute name="Category" use="required">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="autobiography"/>
          <xsd:enumeration value="non-fiction"/>
          <xsd:enumeration value="fiction"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="InStock" type="xsd:boolean" default="false"/>
    <xsd:attribute name="Reviewer" type="xsd:string" default=" "/>
  </xsd:complexType>
</xsd:element>
```

These attributes apply to the element they are nested within (Book)
That is, Book has three attributes – Category, InStock, and Reviewer.

Declaring Attributes Locally

- Notes about inlining
  - The attribute declarations always come last, after the element declarations.
  - The attributes are always with respect to the element that they are defined (nested) within.

"bar "and "boo" are attributes of "foo"

```
<xsd:element name="foo">
   <xsd:complexType>
      <xsd:sequence>
         …
      </xsd:sequence>
      <xsd:attribute name="bar" …/>
      <xsd:attribute name="boo" …/>
   </xsd:complexType>
</xsd:element>
```

Attributes for simpleType Elements

- Intermediate summary: Three ways to declare elements

```
<xsd:element name="name" type="type" minOccurs="int" maxOccurs="int"/>
```

```
<xsd:element name="name" minOccurs="int" maxOccurs="int">
   <xsd:simpleType>
      <xsd:restriction base="type">
      …
      </xsd:restriction>
   </xsd:simpleType>
</xsd:element>
```

```
<xsd:element name="name" minOccurs="int" maxOccurs="int">
      <xsd:complexType>
            …
      </xsd:complexType>
</xsd:element>
```

Attributes for simpleType Elements

comple
xType
simpleT
ype

– allows elements in its content

and

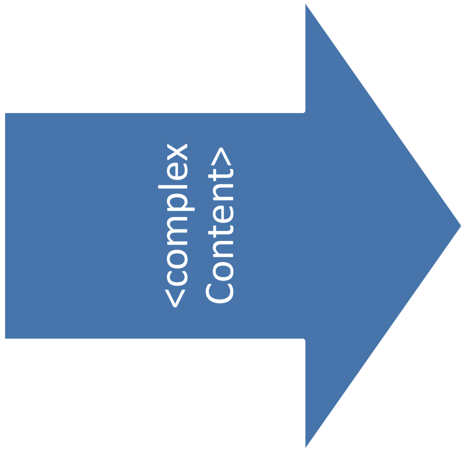
– $\boxed{\text{may carry attributes}}$

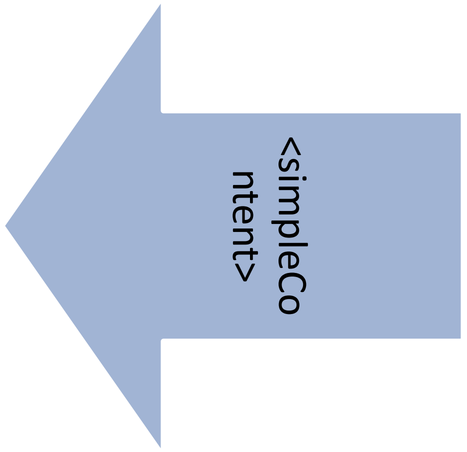
– does not allow elements in its content

and

– may not carry attributes

Attributes for simpleType Elements





- to extend or restrict a complexType

- to extend or restrict a simpleType

Attributes for simpleType Elements

- Element with simple content and attributes
  - Example. Consider this:

```
<elevation units="feet">5440</elevation>
```

  - The elevation element has these two constraints:
    - it has a simple (integer) content
    - it has an attribute called units

© N. Luttenberger

Attributes for simpleType Elements

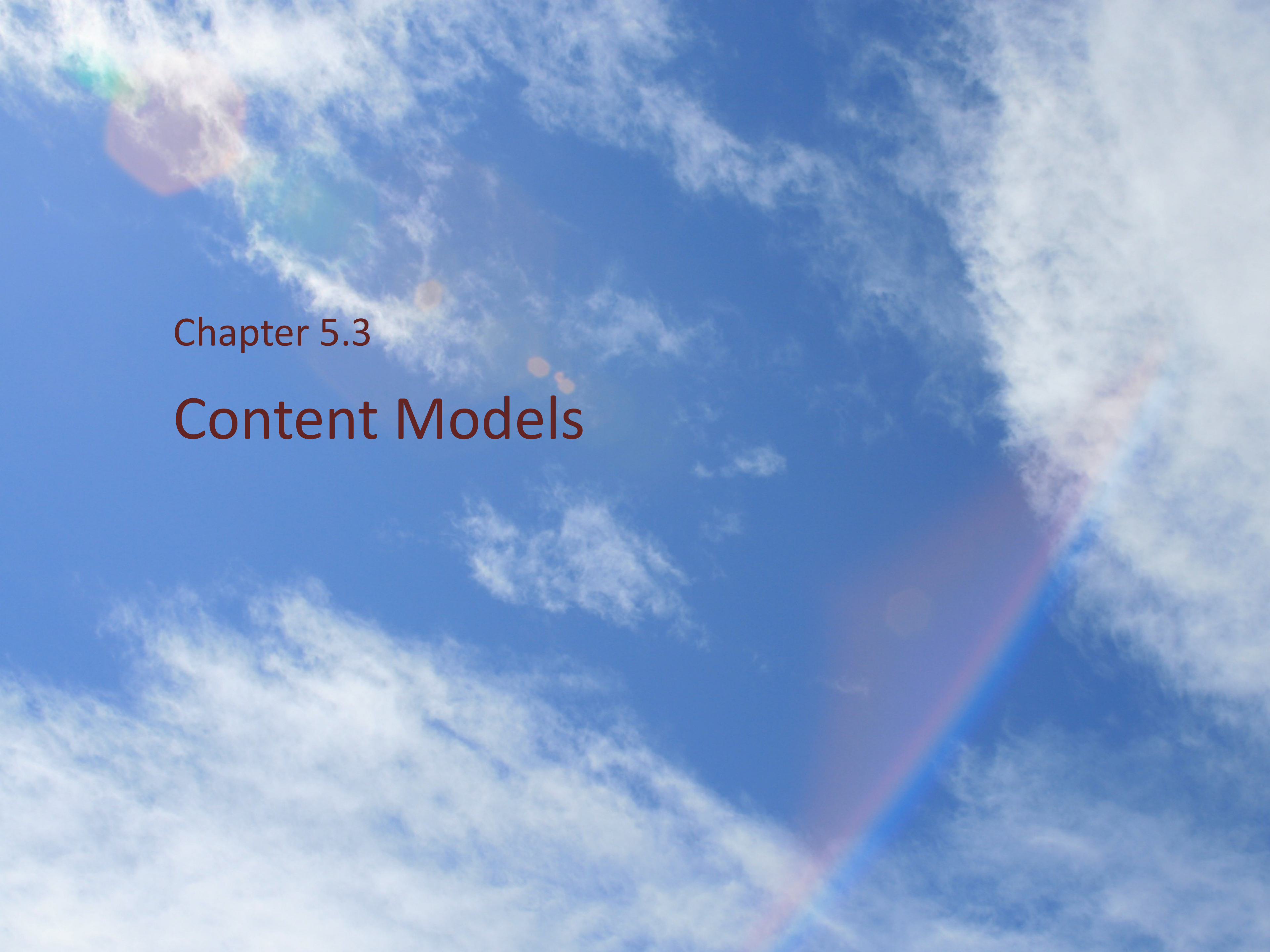
```
<xsd:element name="elevation">
   <xsd:complexType>
      <xsd:simpleContent>
         <xsd:extension base="xsd:integer">
            <xsd:attribute name="units" type="xsd:string"
use="required"/>
         </xsd:extension>
      </xsd:simpleContent>
   </xsd:complexType>
</xsd:element>
```

- elevation contains an attribute: `<xsd:complexType>`
- elevation has no child elements: `<xsd:simpleContent>`
- extend the simpleContent (an integer) with an attribute: `<xsd:extension>`

```
<xsd:simpleType name="elevationRange">
   <xsd:restriction base="xsd:integer">
      <xsd:minInclusive value="0"/>
      <xsd:maxInclusive value="12000"/>
   </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="unitsType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="feet"/>
      <xsd:enumeration value="meters"/>
   </xsd:restriction>
</xsd:simpleType>

<xsd:element name="elevation">
   <xsd:complexType>
      <xsd:simpleContent>
         <xsd:extension base="elevationRange">
            <xsd:attribute name="units" type="unitsType"
use="required"/>
         </xsd:extension>
      </xsd:simpleContent>
   </xsd:complexType>
</xsd:element>
```



Chapter 5.3

Content Models

Element Declaration

comple
xType
simpleT
ype

– allows elements in its content

and

– may carry attributes

– does not allow elements in its content

and

– may not carry attributes

Content Models

- Three kinds of content models
  - sequential (`sequence`):
    Particles in specified order.
  - disjunctive (`choice`):
    Exactly one of the specified particles.
  - conjunctive (`all`):
    all and only exactly zero or one of each particle.
    The particles can occur in any order.
    To reduce implementation complexity,
    only local and top-level element declarations are
    allowed,
    with {min occurs}=0 or 1, {max occurs}=1.

Content Models

- Terminology
  - We call `sequence`, `choice`, `all` *compositors.*
  - Compositors specify the sequence of *element information item children content* in a model group.
  - Model groups occur inside the following Schema elements
    - `complexType`
    - `group`

© N. Luttenberger

Content Models

- `group` element
  - The group element enables you to group together element declarations.
  - Note: the group element is just for grouping together element declarations, no attribute declarations allowed!

```
<xsd:element name="Book" >
   <xsd:complexType>
      <xsd:sequence>
         <xsd:group ref="PublicationElements"/>
         <xsd:element name="ISBN" type="string"/>
         <xsd:element name="Reviewer" type="string"/>
      </xsd:sequence>
   </xsd:complexType>
</xsd:element>
<xsd:element name="CD">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:group ref="PublicationElements"/>
         <xsd:element name="RecordingStudio" type="string"/>
      </xsd:sequence>
   </xsd:complexType>
</xsd:element>
<xsd:group name="PublicationElements">
   <xsd:sequence>
      <xsd:element name="Title" type="xsd:string"/>
      <xsd:element name="Author" type="xsd:string"
maxOccurs="unbounded"/>
      <xsd:element name="Date" type="xsd:string"/>
   </xsd:sequence>
</xsd:group>
```

Content Models

- Group definitions must be global

Cannot inline the group definition.
Instead, you must use a *ref* here and define the group globally.

```
<xsd:element name="Book">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:group name="PublicationElements">
        <xsd:sequence>
          <xsd:element name="Title" type="xsd:string" minOccurs="0"/>
          <xsd:element name="Author" type="xsd:string" minOccurs="0"
                                     maxOccurs="unbounded"/>
          <xsd:element name="Date" type="xsd:string"/>
        </xsd:sequence>
      </xsd:group>
      <xsd:element name="ISBN" type="xsd:string"/>
      <xsd:element name="Publisher" type="xsd:string"/>
    </xsd:sequence>
    ...
  </xsd:complexType>
</xsd:element>
```

Content Models

- Expressing alternates

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                     targetNamespace="http://www.travel.org"
                     xmlns="http://www.travel.org"
                     elementFormDefault="qualified">
  <xsd:element name="transportation">
    <xsd:complexType>
      <xsd:choice>
        <xsd:element name="train" type="xsd:string"/>
        <xsd:element name="plane" type="xsd:string"/>
        <xsd:element name="automobile" type="xsd:string"/>
      </xsd:choice>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Note: the choice is an exclusive-or, that is, transportation can contain only **one** element—either train, or plane, or automobile.

Content Models

- Expressing repeatable choice

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                      targetNamespace="http://www.binary.org"
                      xmlns="http://www.binary.org"
                      elementFormDefault="qualified">
  <xsd:element name="binary-string">
    <xsd:complexType>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element name="zero" type="xsd:unsignedByte" fixed="0"/>
        <xsd:element name="one" type="xsd:unsignedByte" fixed="1"/>
      </xsd:choice>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Content Models

- Using <sequence> and <choice>

```
<?xml version="1.0"?>
<xsd:schema  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
             targetNamespace="http://www.life.org"
             xmlns="http://www.life.org" elementFormDefault="qualified">
  <xsd:element name="life">
    <xsd:complexType>
      <xsd:sequence minOccurs="0" maxOccurs="unbounded">
        <xsd:sequence minOccurs="0" maxOccurs="unbounded">
          <xsd:element name="work" type="xsd:string"/>
          <xsd:element name="eat" type="xsd:string"/>
       </xsd:sequence>
      <xsd:choice>
          <xsd:element name="work" type="xsd:string"/>
          <xsd:element name="play" type="xsd:string"/> </xsd:choice>
        <xsd:element name="sleep" type="xsd:string"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Content Models

- Expressing any order

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                     targetNamespace="http://www.books.org"
                     xmlns="http://www.books.org"
                     elementFormDefault="qualified">
   <xsd:element name="BookStore">
      <xsd:complexType> <xsd:sequence>
            <xsd:element name="Book" maxOccurs="
               <xsd:complexType> <xsd:all>
                     <xsd:element name="Title" type="xsd:string"/>
                     <xsd:element name="Author" type="xsd:string"/>
                     <xsd:element name="Date" type="xsd:string"/>
                     <xsd:element name="ISBN" type="xsd:string"/>
                     <xsd:element name="Publisher" type="xsd:string"/>
                </xsd:all> </xsd:complexType>
            </xsd:element>
      </xsd:sequence> </xsd:complexType>
   </xsd:element>
```

<all> means that Book must contain all five child elements, but they may occur in any order.

Content Models

- Constraints on using <all>
  - Elements declared within <all> must have a maxOccurs value of "1" (minOccurs can be either "0" or "1")
  - If a complexType uses <all> and it extends another type, then that parent type must have empty content.
  - The <all> element cannot be nested within either <sequence>, <choice>, or another <all>
  - The contents of <all> must be just elements. It cannot contain <sequence> or <choice>

Content Models

- Empty element

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                     targetNamespace="http://www.photography.org"
                     xmlns="http://www.photography.org"
                     elementFormDefault="qualified">
  <xsd:element name="gallery">
    <xsd:complexType> <xsd:sequence>
          <xsd:element name="image" maxOccurs="unbounded">
            <xsd:complexType>
              <xsd:attribute name="href" type="xsd:anyURI"
                                   use="required"/>
            </xsd:complexType>
          </xsd:element>
      </xsd:sequence> </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

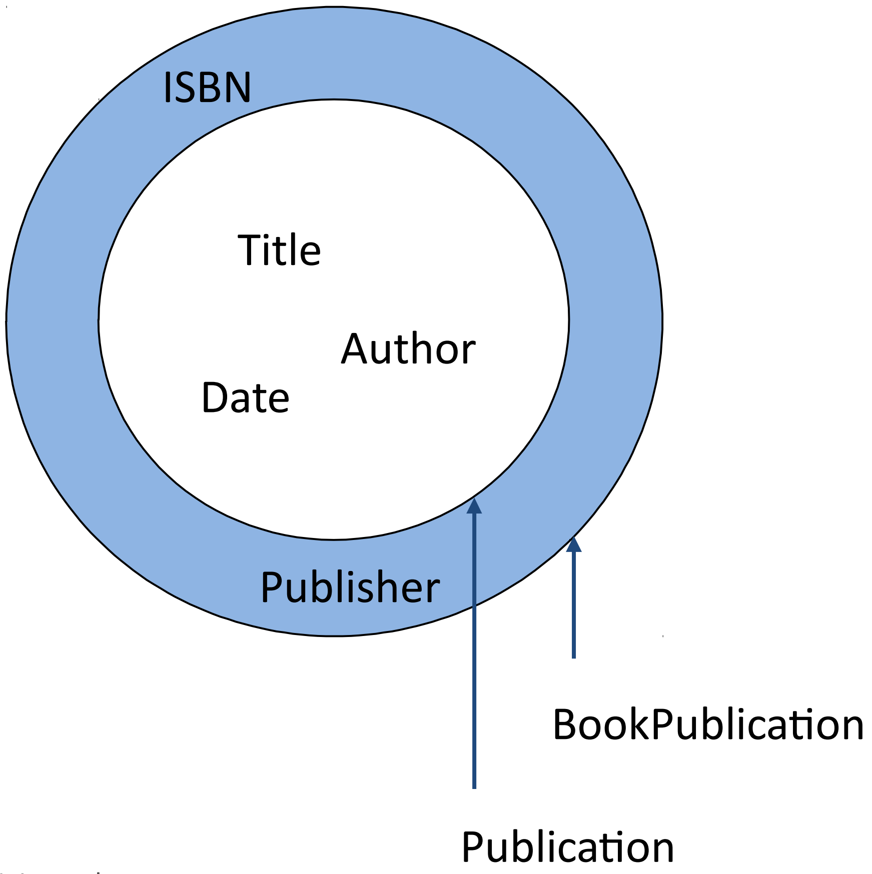
Content Models

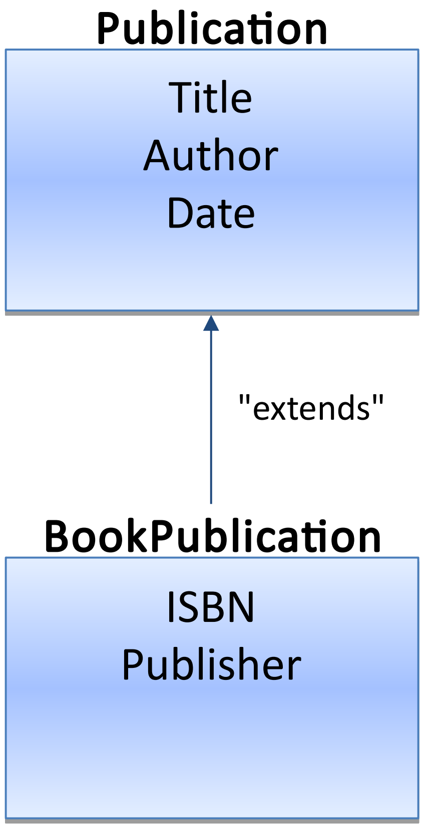
- Subclassing complexType definitions
  - derive by extension:
    extend the parent complexType with more elements.
  - derive by restriction:
    create a type which is a subset of the base type.
    - redefine a base type to have a
      restricted range of values
    - redefine a base type to have a
      more restricted number of occurrences

© N. Luttenberger

Content Models

- Derive by extension





```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                   targetNamespace="http://www.books.org"
                   xmlns="http://www.books.org"
                   elementFormDefault="qualified">
    <xsd:complexType name="Publication">
        <xsd:sequence>
            <xsd:element name="Title" type="xsd:string" maxOccurs="unbounded"/>
            <xsd:element name="Author" type="xsd:string" maxOccurs="unbounded"/>
            <xsd:element name="Date" type="xsd:gYear"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="BookPublication">
        <xsd:complexContent>
            <xsd:extension base="Publication" >
                <xsd:sequence>
                    <xsd:element name="ISBN" type="xsd:string"/>
                    <xsd:element name="Publisher" type="xsd:string"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:element name="BookStore">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="Book" type="BookPubli
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

Note that BookPublication extends the Publication type, i.e., we are doing Derive by Extension

Elements declared to be of type BookPublication will have 5 child elements - Title, Author, Date, ISBN, and Publisher. Note that the elements in the derived type are appended to the elements in the base type.

Content Models

- Derive by restriction

```
<xsd:complexType name="Publication">
   <xsd:sequence>
      <xsd:element name="Title" type="xsd:string" maxOccurs="unbounded"/>
      <xsd:element name="Author" type="xsd:string" maxOccurs="unbounded"/>
      <xsd:element name="Date" type="xsd:gYear"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="SingleAuthorPublication">
   <xsd:complexContent>
      <xsd:restriction base="Publication">
         <xsd:sequence>
            <xsd:element name="Title" type="xsd:string" maxOccurs="unbounded"/>
            <xsd:element name="Author" type="xsd:string"/>
            <xsd:element name="Date" type="xsd:gYear"/>
         </xsd:sequence>
      </xsd:restriction>
   </xsd:complexContent>
</xsd:complexType>
```

There must be exactly one Author element.

In the restriction type you must repeat all the declarations from the base type (exception: see next slide).

Content Models

- Deleting an element in the base type

```
<xsd:complexType name="Publication">
   <xsd:sequence>
      <xsd:element name="Title" type="xsd:string" maxOccurs="unbounded"/>
      <xsd:element name="Author" type="xsd:string" minOccurs="0"/>
      <xsd:element name="Date" type="xsd:gYear"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="ZeroAuthorPublication">
   <xsd:complexContent>
      <xsd:restriction base="Publication">
         <xsd:sequence>
            <xsd:element name="Title" type="xsd:string" maxOccurs="unbounded"/>
            <xsd:element name="Date" type="xsd:gYear"/>
         </xsd:sequence>
      </xsd:restriction>
   </xsd:complexContent>
</xsd:complexType>
```

If the base type has an element with minOccurs="0", and the subtype wishes to not have that element, then it can simply leave it out.

In this subtype we have eliminated the Author element, i.e., the subtype is just comprised of an unbounded number of Title elements followed by a single Date element.

Content Models

- Prohibiting Derivations
  - Sometimes we may want to create a type and
    - disallow all derivations of it, or
    - disallow extension derivations, or
    - disallow restriction derivations.

```
<xsd:complexType name="Publication" final="#all" …>

<xsd:complexType name="Publication" final="extension" …>

<xsd:complexType name="Publication" final="restriction" …>
```

© N. Luttenberger

Chapter 5.4

Element Declaration

Element Declaration

1. Element with simple content

   – Declaring an element using a **built-in** type:

```
<xsd:element name="numStudents" type="xsd:positiveInteger"/>
```

   – Declaring an element using a **user-defined** simpleType:

```
<xsd:simpleType name="shapes">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="triangle"/>
      <xsd:enumeration value="rectangle"/>
      <xsd:enumeration value="square"/>
   </xsd:restriction>
</xsd:simpleType>
<xsd:element name="geometry" type="shapes"/>
```

Element Declaration

2. Element contains child elements
   - Declaring the child elements **inline**

```
<xsd:element name="Person">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element name="Title" type="xsd:string"/>
         <xsd:element name="FirstName" type="xsd:string"/>
         <xsd:element name="Surname" type="xsd:string"/>
      </xsd:sequence>
   </xsd:complexType>
</xsd:element>
```

© N. Luttenberger

Element Declaration

2. Element contains child elements

   – Create a **named complexType**

```
<xsd:complexType name="PersonType">
   <xsd:sequence>
      <xsd:element name="Title" type="xsd:string"/>
      <xsd:element name="FirstName" type="xsd:string"/>
      <xsd:element name="Surname" type="xsd:string"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:element name="Person" type="PersonType"/>
```

© N. Luttenberger

Element Declaration

3. Extension/restriction
   - complexType that is an extension of another complexType

```
<xsd:complexType name="Publication">
   <xsd:sequence>
      <xsd:element name="Title" type="xsd:string" maxOccurs="unbounded"/>
      <xsd:element name="Author" type="xsd:string" maxOccurs="unbounded"/>
      <xsd:element name="Date" type="xsd:gYear"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="BookPublication">
   <xsd:complexContent>
      <xsd:extension base="Publication" >
         <xsd:sequence>
            <xsd:element name="ISBN" type="xsd:string"/>
            <xsd:element name="Publisher" type="xsd:string"/>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
<xsd:element name="Book" type="BookPublication"/>
```

Element Declaration

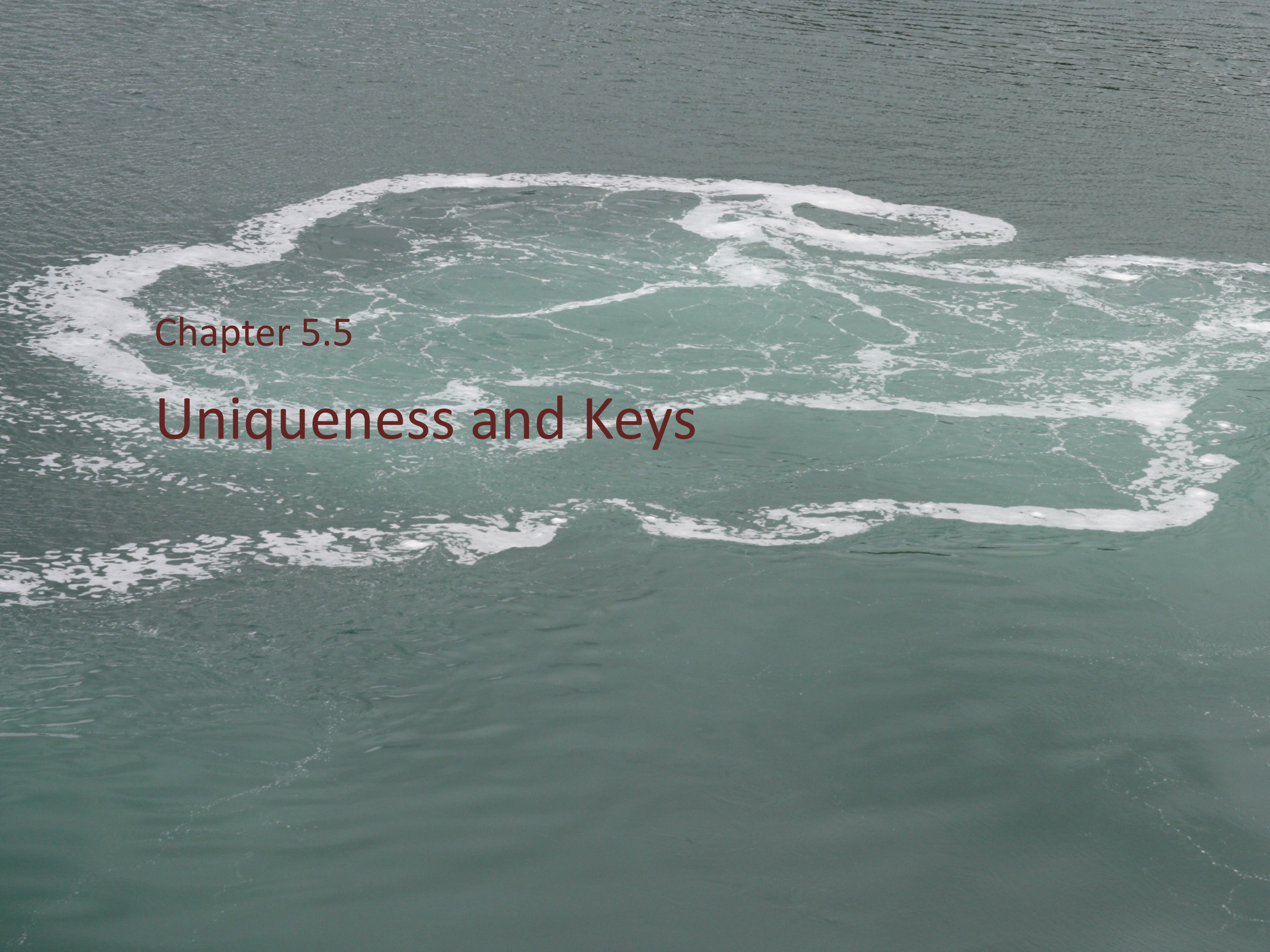
3. Extension/restriction
   - complexType that is a restriction of another complexType

```
<xsd:complexType name="Publication">
   <xsd:sequence>
      <xsd:element name="Title" type="xsd:string" maxOccurs="unbounded"/>
      <xsd:element name="Author" type="xsd:string" maxOccurs="unbounded"/>
      <xsd:element name="Date" type="xsd:gYear"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name= "SingleAuthorPublication">
   <xsd:complexContent>
      <xsd:restriction base="Publication">
         <xsd:sequence>
            <xsd:element name="Title" type="xsd:string" maxOccurs="unbounded"/>
            <xsd:element name="Author" type="xsd:string"/>
            <xsd:element name="Date" type="xsd:gYear"/>
         </xsd:sequence>
      </xsd:restriction>
   </xsd:complexContent>
</xsd:complexType>
<xsd:element name="Catalogue" type="SingleAuthorPublication"/>
```

Element Declaration

4. Element contains simple content and attributes

```
<xsd:element name="apple">
   <xsd:complexType>
      <xsd:simpleContent>
         <xsd:extension base="xsd:string">
            <xsd:attribute name="variety"
                           type="xsd:string" use="required"/>
         </xsd:extension>
      </xsd:simpleContent>
   </xsd:complexType>
</xsd:element>
```



Chapter 5.5

Uniqueness and Keys

Uniqueness & Keys

- XML Schema has enhanced uniqueness capabilities:
  - enables you to define element content to be **unique**.
  - enables you to define non-ID attributes to be unique.
  - enables you to define a combination of element content and attributes to be unique.
  - enables you to distinguish between unique versus **key**.
  - enables you to declare the range of the document over which something is unique.

Key

- Key
  - an element or attribute (or combination thereof) which is defined to be a key must:
    - always be present (minOccurs must be greater than zero)
    - be non-nillable (i.e., nillable="false")
    - have unique content

Key

- Example
  - Within <BookStore>
    each <Book>
    must have an <ISBN>
    and it must be unique.
  - Key is called PK.
    - Select each <Book>, and within each <Book> the ISBN element is a key.

```
<xsd:element name="BookStore">
    ...
    <xsd:key name="PK">
        <xsd:selector xpath="bk:Book"/>
        <xsd:field xpath="bk:ISBN"/>
     </xsd:key>
 </xsd:element>
```

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
           targetNamespace="http://www.books.org"
           xmlns="http://www.books.org"
           xmlns:bk="http://www.books.org"
           elementFormDefault="qualified">
  <xsd:element name="BookStore">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Book" maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="Title" type="xsd:string"/>
              <xsd:element name="Author" type="xsd:string"/>
              <xsd:element name="Date" type="xsd:string"/>
              <xsd:element name="ISBN" type="xsd:string"/>
              <xsd:element name="Publisher" type="xsd:string"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
    <xsd:key name="PK">
       <xsd:selector xpath="bk:Book"/>
       <xsd:field xpath="bk:ISBN"/>
     </xsd:key>
  </xsd:element>
</xsd:schema>
```

```
<?xml version="1.0"?>
<BookStore xmlns="http://www.books.org"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:schemaLocation=
                      "http://www.books.org
                       BookStore.xsd">
    <Book>
        <Title>My Life and Times</Title>
        <Author>Paul McCartney</Author>
        <Date>1998</Date>
        <ISBN>1-56592-235-2</ISBN>
        <Publisher>McMillin Publishing</Publisher>
    </Book>
    <Book>
        <Title>Illusions The Adventures of a Reluctant Messiah</Title>
        <Author>Richard Bach</Author>
        <Date>1977</Date>
        <ISBN>0-440-34319-4</ISBN>
        <Publisher>Dell Publishing Co.</Publisher>
    </Book>
    <Book>
        <Title>The First and Last Freedom</Title>
        <Author>J. Krishnamurti</Author>
        <Date>1954</Date>
        <ISBN>0-06-064831-7</ISBN>
        <Publisher>Harper &amp; Row</Publisher>
    </Book>
</BookStore>
```

A schema-validator will verify that each Book has an ISBN element and that the values are all unique.

Key

- Properties
  - It must be nested within an <element>
  - It must come at the end of <element>
    (after the content model, and attribute declarations)
  - Use the <selector> element as a child of <key> to select a set of elements for which the key applies.
  - Use the <field> element as a child of <key> to identify the element or attribute that is to be the key
  - There can be multiple <field> elements. See next example.

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                targetNamespace="http://www.CostelloReunion.org"
                xmlns="http://www.CostelloReunion.org"
                xmlns:reunion="http://www.CostelloReunion.org"
                elementFormDefault="qualified">
   <xsd:element name="Y2KFamilyReunion">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="Participants"  >
               <xsd:complexType>
                  <xsd:sequence>
                     <xsd:element name="Name" minOccurs="0" maxOccurs="unbounded">
                        <xsd:complexType>
                           <xsd:sequence>
                              <xsd:element name="First" type="xsd:string"/>
                              <xsd:element name="Last" type="xsd:string"/>
                           </xsd:sequence>
                         </xsd:complexType>
                     </xsd:element>
                  </xsd:sequence>
               </xsd:complexType>
            </xsd:element>
         </xsd:sequence>
      </xsd:complexType>
       <xsd:key name="PK">
          <xsd:selector xpath="reunion:Participants/reunion:Name"/>
          <xsd:field xpath="reunion:First"/>
          <xsd:field xpath="reunion:Last"/>
       </xsd:key>
   </xsd:element>
</xsd:schema>
```

unique

- The <unique> element is used exactly like the <key> element is used. It has a <selector> and one or more <field> elements, just like <key> has.
- The only difference is that the schema validator will simply validate that, *whenever present*, the values are unique.

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.books.org"
            xmlns="http://www.books.org"
            xmlns:bk="http://www.books.org"
            elementFormDefault="qualified">
  <xsd:element name="BookStore">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Book" maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="Title" type="xsd:string"/>
              <xsd:element name="Author" type="xsd:string"/>
              <xsd:element name="Date" type="xsd:string"/>
              <xsd:element name="ISBN" type="xsd:string" minOccurs="0"/>
              <xsd:element name="Publisher" type="xsd:string"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
     <xsd:unique name="UNIQ">
        <xsd:selector xpath="bk:Book"/>
        <xsd:field xpath="bk:ISBN"/>
     </xsd:unique>
  </xsd:element>
</xsd:schema>
```

Note: ISBN is optional

Require every ISBN be unique.

```
<?xml version="1.0"?>
<BookStore xmlns="http://www.books.org/namespaces/BookStore"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:schemaLocation=
                     "http://www.books.org/namespaces/BookStore
                      BookStore24.xsd">
    <Book>
        <Title>My Life and Times</Title>
        <Author>Paul McCartney</Author>
        <Date>1998</Date>
        <Publisher>McMillin Publishing</Publisher>
    </Book>
    <Book>
        <Title>Illusions The Adventures of a Reluctant Messiah</Title>
        <Author>Richard Bach</Author>
        <Date>1977</Date>
        <ISBN>0-440-34319-4</ISBN>
        <Publisher>Dell Publishing Co.</Publisher>
    </Book>
    <Book>
        <Title>The First and Last Freedom</Title>
        <Author>J. Krishnamurti</Author>
        <Date>1954</Date>
        <ISBN>0-06-064831-7</ISBN>
        <Publisher>Harper &amp; Row</Publisher>
    </Book>
</BookStore>
```

A schema-validator will verify that each Book which has an ISBN element has a unique value (note that the first Book does not have an ISBN. That's perfectly valid!)

Referencing a key

- Recall that by declaring an element of type IDREF then that element must reference an ID attribute, and an XML Parser will verify that the IDREF value corresponds to a legitimate ID value.

- Similarly, you can define a keyref which asserts, "the value of this element must match the value of an element referred to by this".

```xml
<?xml version="1.0"?>
<Library xmlns="http://www.library.org"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation=
                        "http://www.library.org
                         AuthorSigningAtLibrary.xsd">
   <Books>
      <Book>
            <Title>Illusions The Adventures of a Reluctant Messiah</Title>
            <Author>Richard Bach</Author>
            <Date>1977</Date>
            <ISBN>0-440-34319-4</ISBN>
            <Publisher>Dell Publishing Co.</Publisher>
      </Book>
      ...
   </Books>
   <GuestAuthors>
      <Author>
         <Name>Richard Bach</Name>
         <BookForSigning>
            <Title>Illusions The Adventures of a Reluctant Messiah</Title>
            <ISBN>0-440-34319-4</ISBN>
         </BookForSigning>
      </Author>
   </GuestAuthors>
</Library>
```

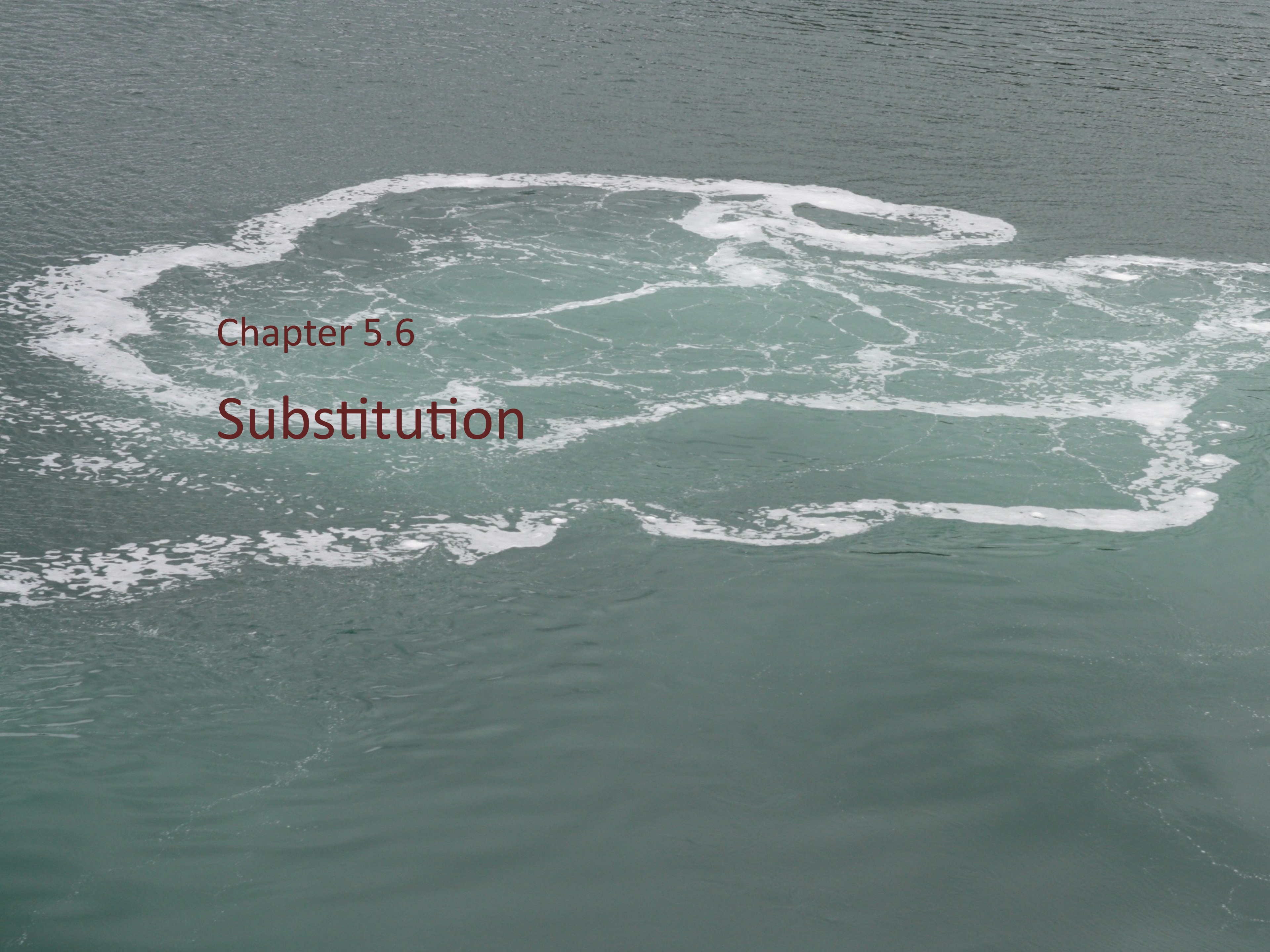
A key element

Suppose each Book
must have an ISBN
and it must be unique.

We would like to ensure
that the ISBN for the
GuestAuthor matches
one of the ISBNs in the
BookStore.

A keyref element

```
<xsd:element name="Library">
      <xsd:complexType>
          <xsd:sequence>
             <xsd:element name="Books">
                <xsd:complexType>
                   <xsd:sequence>
                      <xsd:element ref="Book" maxOccurs="unbounded"/>
                   </xsd:sequence>
                </xsd:complexType>
             </xsd:element>
             <xsd:element ref="GuestAuthors"/>
          </xsd:sequence>
      </xsd:complexType>
      <xsd:key name="PK">
           <xsd:selector xpath="bk:Books/bk:Book"/>
           <xsd:field xpath="bk:ISBN"/>
       </xsd:key>
       <xsd:keyref name="isbnRef" refer="PK">
           <xsd:selector xpath="bk:GuestAuthors/bk:Author/bk:BookForSigning"/>
           <xsd:field xpath="bk:ISBN"/>
       </xsd:keyref>
   </xsd:element>
```



Chapter 5.6

Substitution

Substitution

Element Substitution

- Declare in a Schema
  - an element called "subway",
  - an element called "T",
  - and then state
    that "T"may be substituted for "subway".

  - Instance documents can then use either <subway> or <T>,
    depending on their preference.

Element Substitution

- substitutionGroup

subway is the *head* element

T is substitutable for subway

```
<xsd:element name="subway"  type="xsd:string"/>
<xsd:element name="T"       type="xsd:string"
                            substitutionGroup="subway"/>
```

  – Anywhere a head element can be used in an instance document, any member of the substitutionGroup can be substituted!

Element Substitution

```
<xsd:element name="subway"   type="xsd:string"/>
<xsd:element name="T"        type="xsd:string"
                             substitutionGroup="subway" />
<xsd:element name="transportation">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element ref="subway"/>
      </xsd:sequence>
   </xsd:complexType>
</xsd:element>
```

Schema

```
<transportation>
   <subway>Red Line</subway>
</transportation>
```

Instance doc

```
<transportation>
   <T>Red Line</T>
</transportation>
```

Alternative instance doc

Element Substitution

```
<transportation>
    <subway>Red Line</subway>
</transportation>
```

```
<transporte>
    <metro>Linea Roja</metro>
</transporte>
```

English instance doc

Spanish instance doc

```
<xsd:element name="subway" type="xsd:string"/>
<xsd:element name="transportation"    type="transport"/>
<xsd:complexType name="transport">
   <xsd:sequence>
      <xsd:element ref="subway"/>
   </xsd:sequence>
</xsd:complexType>
```

```
<xsd:element name="transporte" substitutionGroup="transportation"/>
<xsd:element name="metro"       substitutionGroup="subway"/>
```

Element Substitution

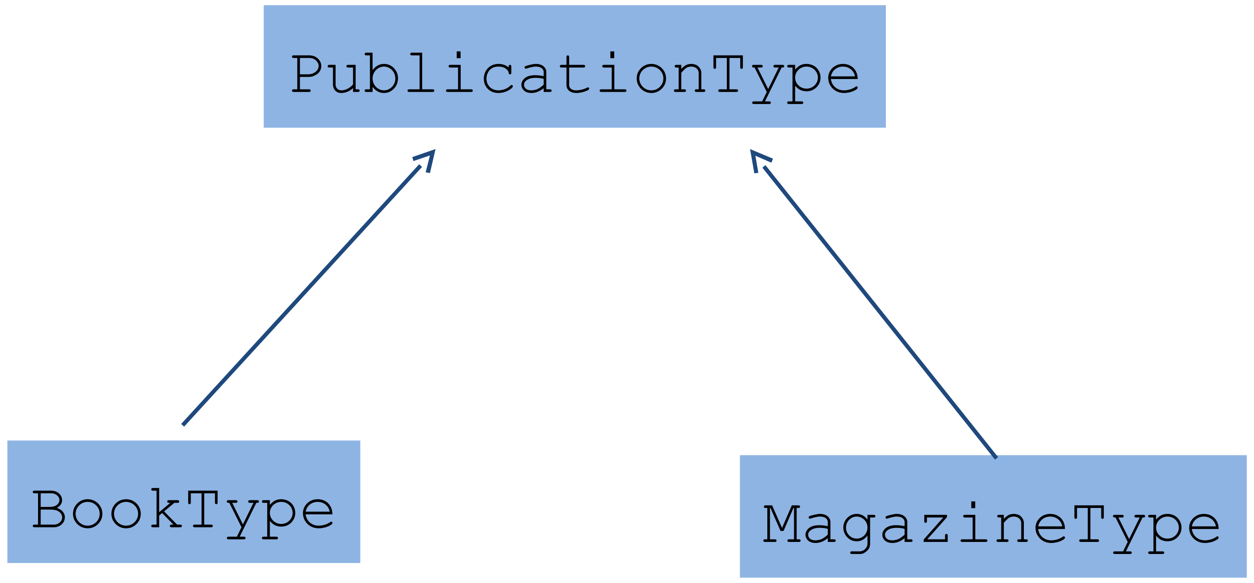
- substitutionGroup element types

```
<xsd:element name="A" type="xxx"/>
<xsd:element name="B" type="yyy" substitutionGroup="A" />
```

This type must be the same as "xxx" or it must be derived from "xxx“.

Element Substitution

- Element Substitution with Derived Types
  - BookType and MagazineType are derived from PublicationType



```
<xsd:complexType name="PublicationType">
  <xsd:sequence>
    <xsd:element name="Title" type="xsd:string"/>
    <xsd:element name="Author" type="xsd:string"
                             minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="Date" type="xsd:gYear"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="BookType">
  <xsd:complexContent>
    <xsd:extension base="PublicationType" >
      <xsd:sequence>
        <xsd:element name="ISBN" type="xsd:string"/>
        <xsd:element name="Publisher" type="xsd:string"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="MagazineType">
  <xsd:complexContent>
    <xsd:restriction base="PublicationType">
      <xsd:sequence>
        <xsd:element name="Title" type="xsd:string"/>
        <xsd:element name="Date" type="xsd:gYear"/>
      </xsd:sequence>
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
```

Element Substitution

- Element Substitution with Derived Types

```
<xsd:element name="Publication" type="PublicationType"/>
<xsd:element name="Book"        type="BookType"
                                substitutionGroup="Publication"/>
<xsd:element name="Magazine"    type="MagazineType"
                                substitutionGroup="Publication"/>
<xsd:element name="BookStore">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element ref="Publication" maxOccurs="unbounded"/>
      </xsd:sequence>
   </xsd:complexType>
</xsd:element>
```

Element Substitution

```
<?xml version="1.0"?>
<BookStore …>
   <Book>
      <Title>Illusions: The Adventures of a Reluctant Messiah</Title>
      <Author>Richard Bach</Author>
      <Date>1977</Date>
      <ISBN>0-440-34319-4</ISBN>
      <Publisher>Dell Publishing Co.</Publisher>
   </Book>
   <Magazine>
      <Title>Natural Health</Title>
      <Date>1999</Date>
   </Magazine>
   <Book>
      <Title>The First and Last Freedom</Title>
      <Author>J. Krishnamurti</Author>
      <Date>1954</Date>
      <ISBN>0-06-064831-7</ISBN>
      <Publisher>Harper &amp; Row</Publisher>
   </Book>
</BookStore>
```

<BookStore> can contain any element in the substitutionGroup with Publication.

Element Substitution

- Blocking element substitution
  - An element may wish to block other elements from substituting with it. This is achieved by adding a block attribute.

```
<xsd:element name="…" type="…" block="substitution"/>
```

Element Substitution

```
<xsd:element name="subway" type="xsd:string" block="substitution"/>
<xsd:element name="T" substitutionGroup="subway"/>
<xsd:element name="transportation">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element ref="subway"/>
      </xsd:sequence>
   </xsd:complexType>
</xsd:element>
```

```
<transportation>
     <subway>Red Line</subway>
</transportation>
```

```
<transportation>
     <T>Red Line</T>
</transportation>
```

Note: there is no error in declaring T to be substitutable with subway. The error occurs only when you try to do substitution in the instance.

Element Substitution

- Rules
  - The elements that are declared to be in the substitution group must be declared as global elements.
  - The type of every element in the substitutionGroup must be the same as, or derived from, the type of the head element.
  - If the type of a substitutionGroup element is the same as the head element then you can omit it.
  - Transitive: If element A can substitute for element B, and element B can substitute for element C, then element A can substitute for element C: If A → B → C then A → C
  - Non-symmetric: If element A can substitute for element B, it is not the case that element B can substitute for element A.

© N. Luttenberger

Type Substitution

- A base type can be substituted by any derived type.
  - Example:
    - Suppose that BookType is derived from PublicationType.
    - We declare an element, Publication, to be of type PublicationType.
    - In the instance document Publication's content can be either of PublicationType or BookType.

© N. Luttenberger

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  targetNamespace="http://www.books.org"
                  xmlns="http://www.books.org"
                  elementFormDefault="unqualified">
   <xsd:complexType name="PublicationType">
      <xsd:sequence>
         <xsd:element name="Title" type="xsd:string"/>
         <xsd:element name="Author" type="xsd:string" maxOccurs="unbounded"/>
         <xsd:element name="Date" type="xsd:year"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="BookType">
      <xsd:complexContent>
         <xsd:extension base="PublicationType">
            <xsd:sequence>
               <xsd:element name="ISBN" type="xsd:string"/>
               <xsd:element name="Publisher" type="xsd:string"/>
            </xsd:sequence>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
   <xsd:element name="BookStore">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="Publication" maxOccurs="unbounded" type="PublicationType"/>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
</xsd:schema>
```

PublicationType is the base type

BookType extends PublicationType

Publication is of type PublicationType (the base type)

```xml
<?xml version="1.0"?>
<bk:BookStore     xmlns:bk="http://www.books.org"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                  xsi:schemaLocation="http://www.books.org BookStore.xsd">
  <Publication>
    <Title>Staying Young Forever</Title>
    <Author>Karin Granstrom Jordan, M.D.</Author>
    <Date>1999</Date>
  </Publication>
  <Publication xsi:type="bk:BookType">
    <Title>Illusions The Adventures of a Reluctant Messiah</Title>
    <Author>Richard Bach</Author>
    <Date>1977</Date>
    <ISBN>0-440-34319-4</ISBN>
    <Publisher>Dell Publishing Co.</Publisher>
  </Publication>
  <Publication xsi:type="bk:BookType">
    <Title>The First and Last Freedom</Title>
    <Author>J. Krishnamurti</Author>
    <Date>1954</Date>
    <ISBN>0-06-064831-7</ISBN>
    <Publisher>Harper &amp; Row</Publisher>
  </Publication>
</bk:BookStore>
```

Type Substitution

- The *type* attribute
  - The Publication element is declared to be of type PublicationType.
  - BookType is derived from PublicationType.
  - As the content is not the source type, but rather a derived type, we specify the derived type that is being used: attribute *type*
  - The attribute *type* comes from the XML Schema Instance (xsi) namespace.

© N. Luttenberger

Chapter 5.7

Abstraction

Abstract Elements

- You can declare an element to be abstract
  - An abstract element is a template/placeholder element.
  - If an element is declared abstract then in an XML instance document that element may not appear.
  - However, elements that are substitutionGroup'ed to the abstract type may appear in its place.

Abstraction

- Abstract elements

```
<xsd:element name="publication" abstract="true"
    type="pubType"/>

<xsd:element name="book" substitutionGrou
    type="pubType"/>

<xsd:element name="magazine" substitutionGroup="publication"
    type="pubType"/>
```

implies: "publication" not allowed in instance doc

- Same mechanisms applicable t

"book" and "magazine" may substitute for "publication"

```
<xsd:complexType name="PublicationType">
   <xsd:sequence>
      <xsd:element name="Title" type="xsd:string"/>
      <xsd:element name="Author" type="xsd:string" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="Date" type="xsd:gYear"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="BookType">
   <xsd:complexContent>
      <xsd:extension base="PublicationType" >
         <xsd:sequence>
            <xsd:element name="ISBN" type="xsd:string"/>
            <xsd:element name="Publisher" type="xsd:string"/>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="MagazineType">
   <xsd:complexContent>
      <xsd:restriction base="PublicationType">
         <xsd:sequence>
            <xsd:element name="Title" type="xsd:string"/>
            <xsd:element name="Author" type="xsd:string" minOcc
            <xsd:element name="Date" type="xsd:gYear"/>
         </xsd:sequence>
      </xsd:restriction>
   </xsd:complexContent>
</xsd:complexType>
<xsd:element name="Publication" abstract="true" type="PublicationType"/>
<xsd:element name="Book" substitutionGroup="Publication" type="BookType"/>
<xsd:element name="Magazine" substitutionGroup="Publication" type="MagazineType"/>
<xsd:element name="BookStore">
   <xsd:complexType> <xsd:sequence>
         <xsd:element ref="Publication" maxOccurs="unbounded"
      </xsd:sequence> </xsd:complexType>
</xsd:element>
```

The Publication element is abstract, only substitutionGroup'ed elements can appear in instance doc.

The Book and Magazine elements are substitutionGroup'ed to the Publication element.

```xml
<?xml version="1.0"?>
<BookStore xmlns="http://www.books.org"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:schemaLocation="http://www.books.org  BookStore.xsd">
   <Magazine>
      <Title>Natural Health</Title>
      <Date>1999</Date>
   </Magazine>
    <Book>
      <Title>Illusions The Adventures of a Reluctant Messiah</Title>
      <Author>Richard Bach</Author>
      <Date>1977</Date>
      <ISBN>0-440-34319-4</ISBN>
      <Publisher>Dell Publishing Co.</Publisher>
   </Book>
   <Book>
      <Title>The First and Last Freedom</Title>
      <Author>J. Krishnamurti</Author>
      <Date>1954</Date>
      <ISBN>0-06-064831-7</ISBN>
      <Publisher>Harper &amp; Row</Publisher>
   </Book>
</BookStore>
```

Abstract Types

- You can declare a complexType to be abstract
  - An abstract complexType is a template/placeholder type.
  - If an element is declared to be a type that is abstract then in an XML instance document the content model of that element may not be that of the abstract type.
  - However, complexType’s that are derived from the abstract type may substitute for the abstract type.

```
<xsd:complexType name="PublicationType" abstract="true">
    <xsd:sequence>
      <xsd:element name="Title" type="xsd:string"  maxOccurs="unbounded"/>
      <xsd:element name="Author" type="xsd:string"  maxOccurs="unbounded"/>
      <xsd:element name="Date" type="xsd:year"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="BookType">
    <xsd:complexContent>
      <xsd:extension base="PublicationType">
        <xsd:sequence>
          <xsd:element name="ISBN" type="xsd:string"/>
          <xsd:element name="Publisher" type="xsd:string"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="SingleAuthorPublication">
    <xsd:complexContent>
      <xsd:restriction base="PublicationType">
        <xsd:sequence>
          <xsd:element name="Title" type="xsd:string"  maxOccurs="unbounded"/>
          <xsd:element name="Author" type="xsd:string"/>
          <xsd:element name="Date" type="xsd:year"/>
        </xsd:sequence>
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="BookStore">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Book"  maxOccurs="unbounded" type="PublicationType"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
```

Note that PublicationType is declared abstract.

Book derives from PublicationType. By default abstract="false". Thus, this type can substitute for the PublicationType.

Abstract Types

```xml
<?xml version="1.0"?>
<BookStore xmlns="http://www.books.org"
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                    xsi:schemaLocation="http://www.books.org BookStore.xsd">
  <Book xsi:type="BookType">
    <Title>My Life and Times</Title>
    <Author>Paul McCartney</Author>
    <Date>1998</Date>
    <ISBN>94303-12021-43892</ISBN>
    <Publisher>McMillin Publishing</Publisher>
  </Book>
  <Book xsi:type="SingleAuthorPublication">
    <Title>FooManchu</Title>
    <Author>Don Keyote</Author>
    <Date>1951</Date>
  </Book>
</BookStore>
```

– The content model of each <Book> element must be from a type that derives from PublicationType.

Abstraction

- If you declare an element to be abstract
  - Use element substitution for the abstract element (as provided by substitutionGroup)
- If you declare a complexType to be abstract
  - Use type substitution for the abstract type (as provided by type derivation)