

INFS 740
DB Programming for the WEB
XML Schema

Professor Alex Brodsky
GMU

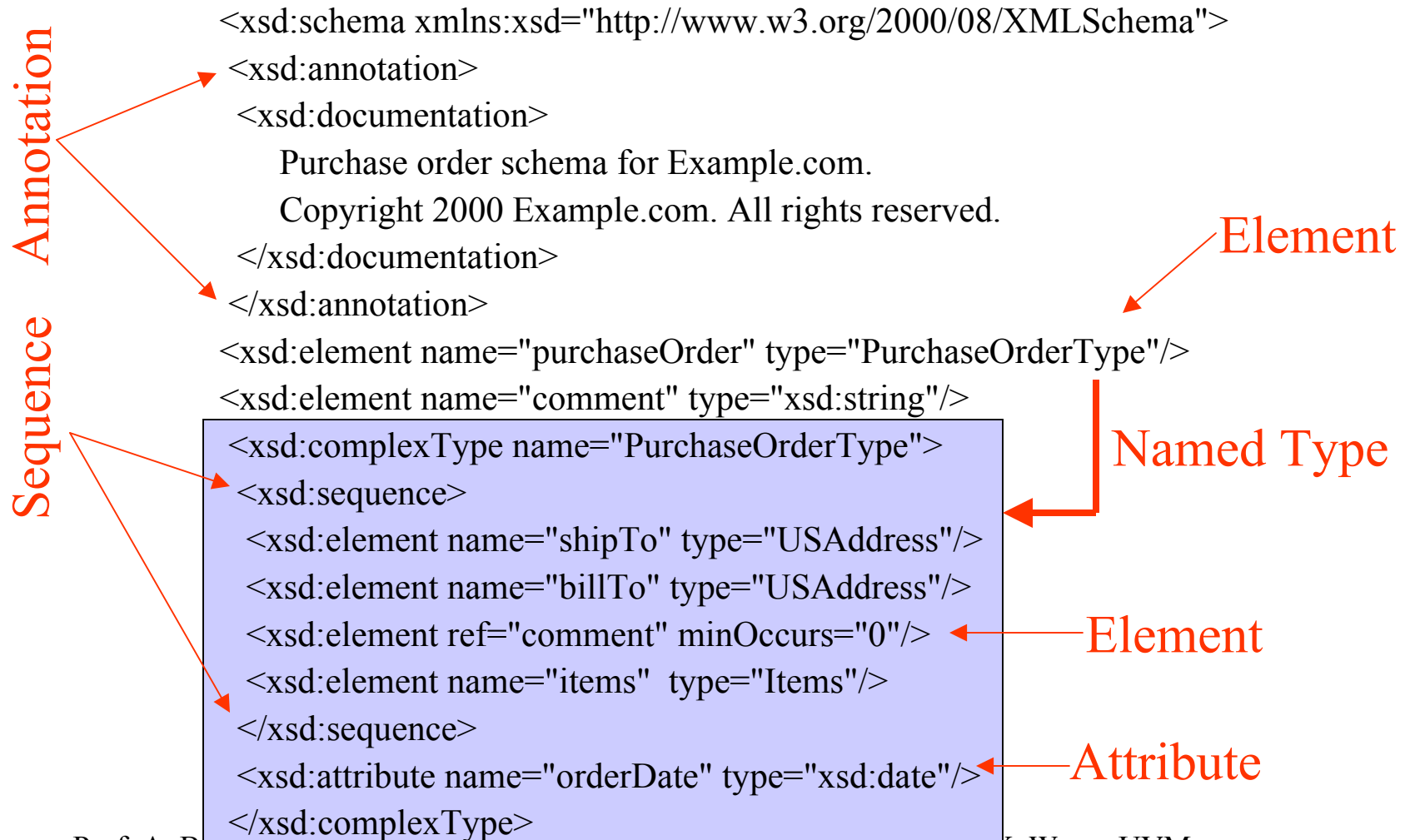
XML Schema PO Example po.xml

```
<?xml version="1.0"?>
<purchaseOrder orderDate="1999-10-20">
  <shipTo country="US">
    <name>Alice Smith</name>
    <street>123 Maple Street</street>
    <city>Mill Valley</city>
    <state>CA</state>
    <zip>90952</zip>
  </shipTo>
  <billTo country="US">
    <name>Robert Smith</name>
    <street>8 Oak Avenue</street>
    <city>Old Town</city>
    <state>PA</state>
    <zip>95819</zip>
  </billTo>
```

po.xml Continued

```
<comment>Hurry, my lawn is going wild!</comment>
<items>
  <item partNum="872-AA">
    <productName>Lawnmower</productName>
    <quantity>1</quantity>
    <USPrice>148.95</USPrice>
    <comment>Confirm this is electric</comment>
  </item>
  <item partNum="926-AA">
    <productName>Baby Monitor</productName>
    <quantity>1</quantity>
    <USPrice>39.98</USPrice>
    <shipDate>1999-05-21</shipDate>
  </item>
</items>
</purchaseOrder>
```

The Purchase Order Schema, po.xsd



po.xsd Continued

```
<xsd:complexType name="USAddress">
  <xsd:sequence>
    <xsd:element name="name" type="xsd:string"/>
    <xsd:element name="street" type="xsd:string"/>
    <xsd:element name="city" type="xsd:string"/>
    <xsd:element name="state" type="xsd:string"/>
    <xsd:element name="zip" type="xsd:decimal"/>
  </xsd:sequence>
  <xsd:attribute name="country" type="xsd:NMTOKEN"
    use="fixed" value="US"/>
</xsd:complexType>
```

po.xsd Continued

```
<xsd:complexType name="Items">
  <xsd:sequence>
    <xsd:element name="item" minOccurs="0" maxOccurs="unbounded">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="productName" type="xsd:string"/>
          <xsd:element name="quantity">
            <xsd:simpleType>
              <xsd:restriction base="xsd:positiveInteger">
                <xsd:maxExclusive value="100"/>
              </xsd:restriction>
            </xsd:simpleType>
          </xsd:element>
          <xsd:element name="USPrice" type="xsd:decimal"/>
          <xsd:element ref="comment" minOccurs="0"/>
          <xsd:element name="shipDate" type="xsd:date" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="partNum" type="SKU"/>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
```

Constraint (points to `<xsd:complexType>`)

Anonymous Element (points to `<xsd:element name="quantity">`)

Constraint (points to `<xsd:restriction base="xsd:positiveInteger">`)

po.xsd Continued

```
</xsd:element> <!-- End item specification -->
</xsd:sequence> <!-- End sequence for items specification -->
</xsd:complexType> <!-- End items specification -->
<!-- Stock Keeping Unit, a code for identifying products -->
<xsd:simpleType name="SKU">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="\d{3}-[A-Z]{2}"/> ← constraint
  </xsd:restriction>
</xsd:simpleType>
</xsd:schema>
```

Main Schema Components

Definitions of:

- Complex types - sub-elements + attributes
- Simple types - no sub-elements, constraints on strings (datatypes)

Declarations of:

- elements (of simple and complex types)
- attributes (simple types), attribute groups

Declaration vs Definition

- *declarations* - things that will be used in an XML instance document.
 - element declarations
 - attribute declarations
- *definitions* - things that are just used in the schema document; a definition creates a new type
 - simple type definitions
 - complex type definitions
 - attribute group, model group definitions

Constraints on Element Content

Type of content

- textOnly - only character data
- elementOnly - only subelements
- mixed - character data appears alongside subelements
- any – anything goes
- empty - no content (only attributes)

```
<element name="cost">  
  <complexType content="empty">  
    <attribute name="currency" type="string"/>  
    <attribute name="value" type="decimal"/>  
  </complexType>  
</element>
```

```
XML: <cost currency="USD" value="25.70"/>
```

XSD Complex Types Indicators

- Order Indicators:
 - *all* – appear in any order, but must all appear
 - *sequence* – all must appear, and in the given order (concatenation)
 - *choice* – may pick any one, but only one

Examples

```
<xs:complexType>  
  <xs:sequence>  
    <xs:element name="full_name" type="xs:string"/>  
    <xs:element name="child_name" type="xs:string" maxOccurs="10"  
      minOccurs="0"/>  
  </xs:sequence>  
</xs:complexType>
```

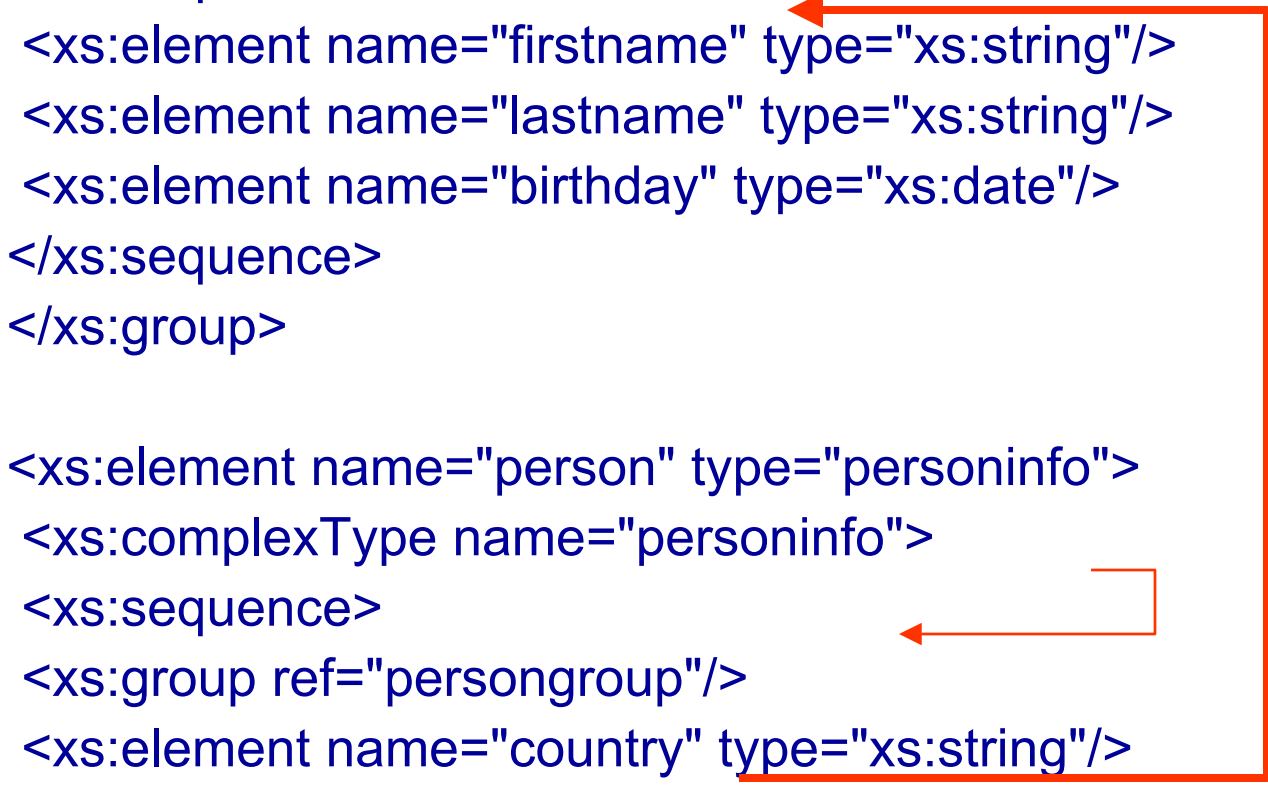
```
<xs:complexType>  
  <xs:all>  
    <xs:element name="full_name" type="xs:string"/>  
    <xs:element name="child_name" type="xs:string" maxOccurs="10"  
      minOccurs="0"/>  
  </xs:all>  
</xs:complexType>
```

Occurrence Indicator

- maxOccurs
- minOccurs

Group Indicator

```
<xs:group name="persongroup">  
  <xs:sequence>  
    <xs:element name="firstname" type="xs:string"/>  
    <xs:element name="lastname" type="xs:string"/>  
    <xs:element name="birthday" type="xs:date"/>  
  </xs:sequence>  
</xs:group>
```

A red rectangular box encloses the first XML snippet. A red arrow originates from the top-right corner of the box and points to the left, ending at the opening tag of the second XML snippet.

```
<xs:element name="person" type="personinfo">  
  <xs:complexType name="personinfo">  
    <xs:sequence>  
      <xs:group ref="persongroup"/>  
      <xs:element name="country" type="xs:string"/>  
    </xs:sequence>  
  </xs:complexType>  
</xs:element>
```

Attribute Group

```
<xs:attributeGroup name="personattrgroup">  
  <xs:attribute name="firstname" type="xs:string"/>  
  <xs:attribute name="lastname" type="xs:string"/>  
  <xs:attribute name="birthday" type="xs:date"/>  
</xs:attributeGroup>  
  
<xs:element name="person">  
  <xs:complexType>  
    <xs:attributeGroup ref="personattrgroup"/>  
  </xs:complexType>  
</xs:element>
```

Derived Types

Subclassing of type definitions

- derive by *extension*: extend the parent type with more elements
- derive by *restriction*: constrain the parent type by constraining some of the elements to have a more restricted range of values, or a more restricted number of occurrences.
- derive by *reproduction* - copy of the parent type

Deriving Types by Extension

- extend a base type's content by adding elements and/or attributes

```
<complexType name="newType" base="baseType"  
              derivedBy="extension"/>  
  <element name="newelt" type="string"/>  
  <attribute name="newattr" type="integer"/>  
</complexType>
```

Deriving Types by Restriction

- Narrows the ranges or reduces alternatives of elements and/or attributes

```
<complexType name="myType">
  <element name="elt1" type="someType" minOccurs="0"/>
  <attribute name="attr1" type="string"/>
</complexType>

<complexType name="restrictedType" base="myType"
              derivedBy="restriction">
  <element name="elt1" type="someType" minOccurs="1"/>
  <attribute name="attr1" type="string" fixed="sausage"/>
</complexType>
```

Preventing Type Derivations

- There are mechanisms to prevent type derivations from a particular type definition

Uniqueness

XML Schemas can specify:

- any attribute or element to be unique
- combinations of attribute and element values to be unique
- the range of the document over which something is unique

Defining Uniqueness

Need to specify

- a selector which is an Xpath expression to a parent element
- one or more fields - Xpath expressions to subelements or attributes within the selector

```
<unique name="newsUnique">  
  <selector>VideoCatalogue/NewsCatalogue/NewsDoc</selector>  
  <field>Title</field>  
  <field>@Category</field>  
</unique>
```

The combination of the contents of Title plus the value of the attribute Category must be unique throughout an XML instance document. The Title element and Category attribute lie within the XPath expression shown in the selector element.

unique vs key

- Key: an element/attribute (or combination thereof) which is defined to be a key must
 - always be present (minOccurs must be greater than zero)
 - be non-nullable (i.e., nullable="false")
 - be unique
- Key implies unique, but unique does not imply key

Defining a Key

Need to specify

- a selector which is an Xpath expression to a parent element
- fields - Xpath expressions to subelements or attributes

```
<key name="newsKey">
  <selector>VideoCatalogue/NewsCatalogue/NewsDoc</selector>
  <field>Title</field>
  <field>@Category</field>
</key>
```

The combination of the contents of Title plus the value of the attribute Category is to be unique throughout an XML instance document. The Title element and Category attribute lie within the XPath expression shown in the selector element.