CSIT884:

Web Development

XML Schema Definition - XSD

School of Computing and Information Technology

University of Wollongong

XSD

- **X**ML **S**chema **D**efinition (XSD) is another way to define the legal building blocks of an XML document. It defines the document structure with a list of legal elements and attributes.

- Using a XSD, different parties can agree on a standard XML format for interchanging data.

- We can check whether an XML document conforms to a XSD or not.

- File extension is .xsd

XSD: student example

XML file:

```
<?xml version="1.0" ?>
<student xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="student.xsd">
  <firstName>John</firstName>
  <lastName>Smith</lastName>
  <email>jsmith@gmail.com</email>
  <mobile>0211223344</mobile>
</student>
```

XSD file student.xsd:

```
<?xml version="1.0" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

 <xsd:element name="student">
   <xsd:complexType>
     <xsd:sequence>
       <xsd:element name="firstName" type="xsd:string"/>
       <xsd:element name="lastName" type="xsd:string"/>
       <xsd:element name="email" type="xsd:string"/>
       <xsd:element name="mobile" type="xsd:string"/>
     </xsd:sequence>
   </xsd:complexType>
 </xsd:element>

</xsd:schema>
```

XSD: `student` example

XML file:

```
<?xml version="1.0" ?>
<student xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="student.xsd">
  <firstName>John</firstName>
  <lastName>Smith</lastName>
  <email>jsmith@gmail.com</email>
  <mobile>0211223344</mobile>
</student>
```

elements and data types used in the schema come from the namespace http://www.w3.org/2001/XMLSchema

XSD file **`student.xsd`**:

```
<?xml version="1.0" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

 <xsd:element name="student">
   <xsd:complexType>
     <xsd:sequence>
       <xsd:element name="firstName" type="xsd:string"/>
       <xsd:element name="lastName" type="xsd:string"/>
       <xsd:element name="email" type="xsd:string"/>
       <xsd:element name="mobile" type="xsd:string"/>
     </xsd:sequence>
   </xsd:complexType>
 </xsd:element>

</xsd:schema>
```

XSD: student example

XML file:

```
<?xml version="1.0" ?>
<student xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="student.xsd">
  <firstName>John</firstName>
  <lastName>Smith</lastName>
  <email>jsmith@gmail.com</email>
  <mobile>0211223344</mobile>
</student>
```

the elements and data types that come from the namespace http://www.w3.org/2001/XMLSchema should be prefixed with xsd

XSD file **student.xsd**:

```
<?xml version="1.0" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

 <xsd:element name="student">
   <xsd:complexType>
     <xsd:sequence>
       <xsd:element name="firstName" type="xsd:string"/>
       <xsd:element name="lastName" type="xsd:string"/>
       <xsd:element name="email" type="xsd:string"/>
       <xsd:element name="mobile" type="xsd:string"/>
     </xsd:sequence>
   </xsd:complexType>
 </xsd:element>

</xsd:schema>
```

XSD: `student` example

XML file:

```
<?xml version="1.0" ?>
<student xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="student.xsd">
  <firstName>John</firstName>
  <lastName>Smith</lastName>
  <email>jsmith@gmail.com</email>
  <mobile>0211223344</mobile>
</student>
```

XSD file `student.xsd`:

```
<?xml version="1.0" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

 <xsd:element name="student">
   <xsd:complexType>
     <xsd:sequence>
       <xsd:element name="firstName" type="xsd:string"/>
       <xsd:element name="lastName" type="xsd:string"/>
       <xsd:element name="email" type="xsd:string"/>
       <xsd:element name="mobile" type="xsd:string"/>
     </xsd:sequence>
   </xsd:complexType>
 </xsd:element>

</xsd:schema>
```

Complex type ?

Simple type ?

XSD: element

XML element can be defined in XSD as 2 types:

- `simpleType`
- `complexType`

- Element contains NO attributes, NO elements → `simpleType`
- Element contains other elements → `complexType`
- Element contains attributes → `complexType`

XSD: simple type containing no element, no attribute

- Element contains no elements, no attributes → `simpleType`

xml file

```
<website>http://www.uow.edu.au/student</website> . . . (1)

<lastDayToEnrol>2000-03-24</lastDayToEnrol> . . . . . (2)

<favouriteColor>blue</favouriteColor> . . . . . . . . (3)
```

xsd file

```
<xsd:element name="website" type="xsd:anyURI" /> . . . . . (1)

<xsd:element name="lastDayToEnrol" type="xsd:date" /> . . . (2)

<xsd:element name="favouriteColor" type="xsd:string" /> . . (3)
```

XSD: simple type containing no element, no attribute

- Simple elements may have a default value, or a fixed value

```
<xsd:element name="customer" type="xsd:string" default="unknown" />

<xsd:element name="colour" type="xsd:string fixed="red" />
```

XSD: simple type with restriction

Grade can have one of the 4 values: A, B, C, D

```
<grade>B</grade>
```

Without restriction:

```
<xsd:element name="grade" type="xsd:string" />
```

With restriction:

```
<xsd:element name="grade">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="A"/>
      <xsd:enumeration value="B"/>
      <xsd:enumeration value="C"/>
      <xsd:enumeration value="D"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

XSD: simple type with restriction

Grade can have one of the 4 values: A, B, C, D

```
<grade>B</grade>
```

```
<xsd:element name="grade">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="[ABCD]"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

XSD: simple type with restriction

letter element - acceptable value is one of the lowercase letters from a to z

```
<letter>x</letter>
```

```
<xsd:element name="letter">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="[a-z]"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

XSD: simple type with restriction

Mark can have values between 0-100

```
<mark>84</mark>
```

Without restriction:

```
<xsd:element name="mark" type="xsd:integer" />
```

With restriction:

```
<xsd:element name="mark">
  <xsd:simpleType>
    <xsd:restriction base="xsd:integer">
      <xsd:minInclusive value="0"/>
      <xsd:maxInclusive value="100"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

XSD: constraining facets

Constraint	Description
enumeration	Defines a list of acceptable values
fractionDigits	Specifies the maximum number of decimal places allowed. Must be equal to or greater than zero
length	Specifies the exact number of characters or list items allowed. Must be equal to or greater than zero
maxExclusive	Specifies the upper bounds for numeric values (the value must be less than this value)
maxInclusive	Specifies the upper bounds for numeric values (the value must be less than or equal to this value)
maxLength	Specifies the maximum number of characters or list items allowed. Must be equal to or greater than zero
minExclusive	Specifies the lower bounds for numeric values (the value must be greater than this value)
minInclusive	Specifies the lower bounds for numeric values (the value must be greater than or equal to this value)
minLength	Specifies the minimum number of characters or list items allowed. Must be equal to or greater than zero
pattern	Defines the exact sequence of characters that are acceptable
totalDigits	Specifies the exact number of digits allowed. Must be greater than zero
whiteSpace	Specifies how white space (line feeds, tabs, spaces, and carriage returns) is handled

XSD: attributes

- The attribute itself is always declared as a simple type

xml file

```
<lastname lang="EN">Smith</lastname>
```

xsd file

```
. . .
<xsd:attribute name="lang" type="xs:string"/> . . . (1)
```

OR

```
<xsd:attribute name="lang" type="xsd:string" use="required" />. . (2)
```

OR

```
<xsd:attribute name="lang" type="xsd:string" default="EN" /> . . (3)
```

OR

```
<xsd:attribute name="lang" type="xsd:string" fixed="EN" /> . . (4)
```

XSD: complex type containing element

- Element contains other elements → `complexType`

```
<result>
  <mark>85</mark>
  <grade>A</grade>
</result>
```

```
<xsd:element name="result">
  <xsd:complexType>
    <xsd:sequence>
       <xsd:element name="mark" type="xsd:integer"/>
       <xsd:element name="grade" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

XSD: complex type containing element and attribute

- Element contains other elements and attributes → `complexType`

xml file

```
<scan schedule="hourly">
  <start>2018-06-20T13:00:00</start>
  <finish>2018-06-20T13:01:47</finish>
  <virusFound>true</virusFound>
</scan>
```

The attribute declarations must always come last

xsd file

```
<xsd:element name="scan">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="start" type="xsd:dateTime" />
      <xsd:element name="finish" type="xsd:dateTime" />
      <xsd:element name="virusFound" type="xsd:boolean" />
    </xsd:sequence>
    <xsd:attribute name="schedule" type="xsd:string" />
  </xsd:complexType>
</xsd:element>
```

XSD: complex type containing attributes only

- Text-only element contains attributes (*does not contain elements*)
  → `complexType`

xml file

```
<price promotionCode="FAMILYDEAL">39.50</price>
```

xsd file

```
<xsd:element name="price">
  <xsd:complexType>
   <xsd:simpleContent>
     <xsd:extension base="xsd:decimal">
       <xsd:attribute name="promotionCode" type="xsd:string" />
     </xsd:extension>
   </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
```

XSD: studentList example

xml file

```
<?xml version="1.0" ?>
<studentList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="studentList.xsd">
  <student>
    <firstName>John</firstName>
    <lastName>Smith</lastName>
    <email>jsmith@gmail.com</email>
  </student>
  <student>
    <firstName>Mary</firstName>
    <lastName>Jane</lastName>
    <email>mjane@gmail.com</email>
  </student>
</studentList>
```

xsd file

```
<?xml version="1.0" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="studentList">
   <xsd:complexType>
     <xsd:sequence>
       <xsd:element name="student" minOccurs="0" maxOccurs="unbounded">
         <xsd:complexType>
           <xsd:sequence>
             <xsd:element name="firstName" type="xsd:string"/>
             <xsd:element name="lastName" type="xsd:string"/>
             <xsd:element name="email" type="xsd:string"/>
           </xsd:sequence>
         </xsd:complexType>
       </xsd:element>
     </xsd:sequence>
   </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

XSD: studentList example

```
<?xml version="1.0" ?>
<studentList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="studentList.xsd">
  <student>
    <firstName>John</firstName>
    <lastName>Smith</lastName>
    <email>jsmith@gmail.com</email>
  </student>
  <student>
    <firstName>Mary</firstName>
    <lastName>Jane</lastName>
    <email>mjane@gmail.com</email>
  </student>
</studentList>
```

Let’s start with the root element **studentList**

xsd file

```
<?xml version="1.0" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="studentList">
   <xsd:complexType>
     <xsd:sequence>
       <xsd:element name="student" minOccurs="0" maxOccurs="unbounded">
         <xsd:complexType>
           <xsd:sequence>
             <xsd:element name="firstName" type="xsd:string"/>
             <xsd:element name="lastName" type="xsd:string"/>
             <xsd:element name="email" type="xsd:string"/>
           </xsd:sequence>
         </xsd:complexType>
       </xsd:element>
     </xsd:sequence>
   </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

XSD: studentList example

```
<?xml version="1.0" ?>
<studentList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="studentList.xsd">
  <student>
    <firstName>John</firstName>
    <lastName>Smith</lastName>
    <email>jsmith@gmail.com</email>
  </student>
  <student>
    <firstName>Mary</firstName>
    <lastName>Jane</lastName>
    <email>mjane@gmail.com</email>
  </student>
</studentList>
```

Let's start with the root element **studentList**

- it is a complex type

```
<?xml version="1.0" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="studentList">
   <xsd:complexType>
     <xsd:sequence>
       <xsd:element name="student" minOccurs="0" maxOccurs="unbounded">
         <xsd:complexType>
           <xsd:sequence>
             <xsd:element name="firstName" type="xsd:string"/>
             <xsd:element name="lastName" type="xsd:string"/>
             <xsd:element name="email" type="xsd:string"/>
           </xsd:sequence>
         </xsd:complexType>
       </xsd:element>
     </xsd:sequence>
   </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

XSD: studentList example

```
<?xml version="1.0" ?>
<studentList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="studentList.xsd">
  <student>
    <firstName>John</firstName>
    <lastName>Smith</lastName>
    <email>jsmith@gmail.com</email>
  </student>
  <student>
    <firstName>Mary</firstName>
    <lastName>Jane</lastName>
    <email>mjane@gmail.com</email>
  </student>
</studentList>
```

Let’s start with the root element **studentList**

- it is a complex type
- which contains a sequence of **student** elements

```
<?xml version="1.0" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="studentList">
   <xsd:complexType>
     <xsd:sequence>
       <xsd:element name="student" minOccurs="0" maxOccurs="unbounded">
         <xsd:complexType>
           <xsd:sequence>
             <xsd:element name="firstName" type="xsd:string"/>
             <xsd:element name="lastName" type="xsd:string"/>
             <xsd:element name="email" type="xsd:string"/>
           </xsd:sequence>
         </xsd:complexType>
       </xsd:element>
     </xsd:sequence>
   </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

XSD: studentList example

```
<?xml version="1.0" ?>
<studentList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="studentList.xsd">
  <student>
    <firstName>John</firstName>
    <lastName>Smith</lastName>
    <email>jsmith@gmail.com</email>
  </student>
  <student>
    <firstName>Mary</firstName>
    <lastName>Jane</lastName>
    <email>mjane@gmail.com</email>
  </student>
</studentList>
```

Let’s start with the root element **studentList**

- it is a complex type
- which contains a sequence of **student** elements
- **studentList** contains **zero** or **unlimited** number of **student** elements

```
<?xml version="1.0" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="studentList">
   <xsd:complexType>
     <xsd:sequence>
       <xsd:element name="student" minOccurs="0" maxOccurs="unbounded">
         <xsd:complexType>
           <xsd:sequence>
             <xsd:element name="firstName" type="xsd:string"/>
             <xsd:element name="lastName" type="xsd:string"/>
             <xsd:element name="email" type="xsd:string"/>
           </xsd:sequence>
         </xsd:complexType>
       </xsd:element>
     </xsd:sequence>
   </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

XSD: studentList example

```
<?xml version="1.0" ?>
<studentList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="studentList.xsd">
  <student>
    <firstName>John</firstName>
    <lastName>Smith</lastName>
    <email>jsmith@gmail.com</email>
  </student>
  <student>
    <firstName>Mary</firstName>
    <lastName>Jane</lastName>
    <email>mjane@gmail.com</email>
  </student>
</studentList>
```

The element **student** is also a complex type

```
<?xml version="1.0" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="studentList">
   <xsd:complexType>
     <xsd:sequence>
       <xsd:element name="student" minOccurs="0" maxOccurs="unbounded">
         <xsd:complexType>
           <xsd:sequence>
             <xsd:element name="firstName" type="xsd:string"/>
             <xsd:element name="lastName" type="xsd:string"/>
             <xsd:element name="email" type="xsd:string"/>
           </xsd:sequence>
         </xsd:complexType>
       </xsd:element>
     </xsd:sequence>
   </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

XSD: studentList example

```
<?xml version="1.0" ?>
<studentList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="studentList.xsd">
  <student>
    <firstName>John</firstName>
    <lastName>Smith</lastName>
    <email>jsmith@gmail.com</email>
  </student>
  <student>
    <firstName>Mary</firstName>
    <lastName>Jane</lastName>
    <email>mjane@gmail.com</email>
  </student>
</studentList>
```

The element **student** is also a complex type

- which contains a sequence of elements

```
<?xml version="1.0" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="studentList">
   <xsd:complexType>
     <xsd:sequence>
       <xsd:element name="student" minOccurs="0" maxOccurs="unbounded">
         <xsd:complexType>
           <xsd:sequence>
             <xsd:element name="firstName" type="xsd:string"/>
             <xsd:element name="lastName" type="xsd:string"/>
             <xsd:element name="email" type="xsd:string"/>
           </xsd:sequence>
         </xsd:complexType>
       </xsd:element>
     </xsd:sequence>
   </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

XSD: studentList example

```
<?xml version="1.0" ?>
<studentList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="studentList.xsd">
  <student>
    <firstName>John</firstName>
    <lastName>Smith</lastName>
    <email>jsmith@gmail.com</email>
  </student>
  <student>
    <firstName>Mary</firstName>
    <lastName>Jane</lastName>
    <email>mjane@gmail.com</email>
  </student>
</studentList>
```

The element **student** is also a complex type

- which contains a sequence of elements: **firstName, lastName, email**

```
<?xml version="1.0" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="studentList">
   <xsd:complexType>
     <xsd:sequence>
       <xsd:element name="student" minOccurs="0" maxOccurs="unbounded">
         <xsd:complexType>
           <xsd:sequence>
             <xsd:element name="firstName" type="xsd:string"/>
             <xsd:element name="lastName" type="xsd:string"/>
             <xsd:element name="email" type="xsd:string"/>
           </xsd:sequence>
         </xsd:complexType>
       </xsd:element>
     </xsd:sequence>
   </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

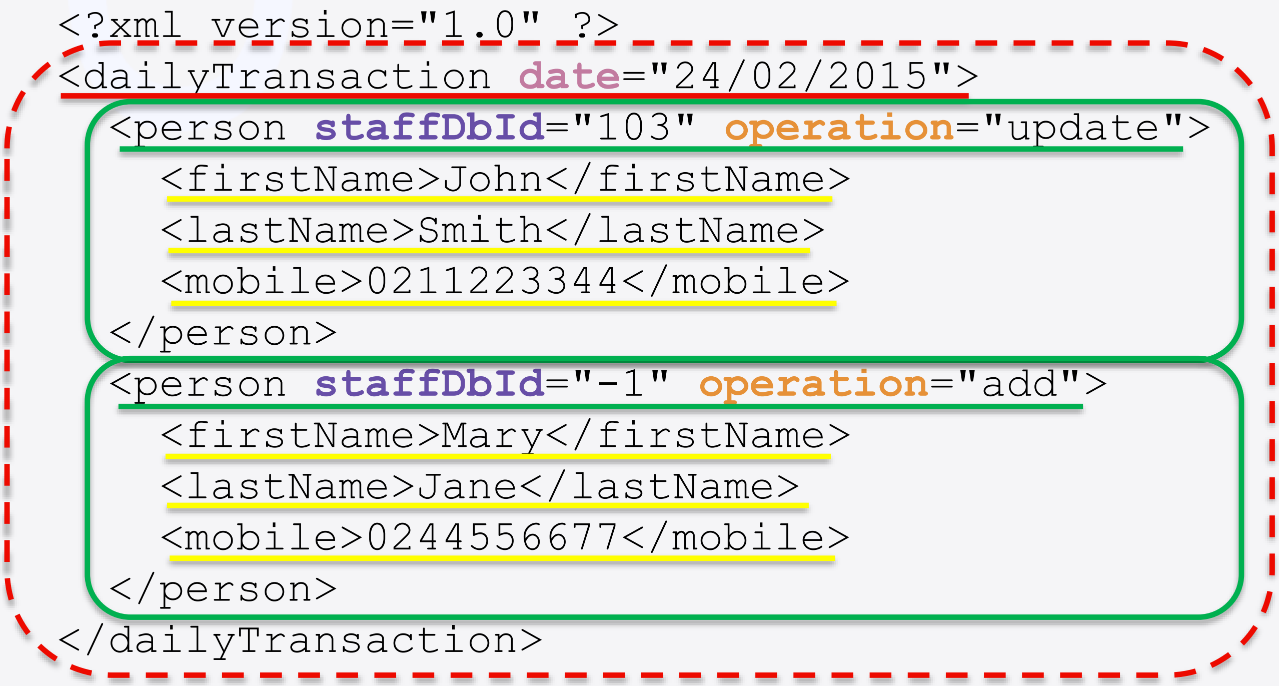
XSD: studentList example

```
<?xml version="1.0" ?>
<studentList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="studentList.xsd">
  <student>
    <firstName>John</firstName>
    <lastName>Smith</lastName>
    <email>jsmith@gmail.com</email>
  </student>
  <student>
    <firstName>Mary</firstName>
    <lastName>Jane</lastName>
    <email>mjane@gmail.com</email>
  </student>
</studentList>
```

firstName, lastName, email elements are all simple type

```
<?xml version="1.0" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="studentList">
   <xsd:complexType>
     <xsd:sequence>
       <xsd:element name="student" minOccurs="0" maxOccurs="unbounded">
         <xsd:complexType>
           <xsd:sequence>
             <xsd:element name="firstName" type="xsd:string"/>
             <xsd:element name="lastName" type="xsd:string"/>
             <xsd:element name="email" type="xsd:string"/>
           </xsd:sequence>
         </xsd:complexType>
       </xsd:element>
     </xsd:sequence>
   </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

XSD: dailyTransaction example



```
<?xml version="1.0" ?>
<dailyTransaction date="24/02/2015">
  <person staffDbId="103" operation="update">
    <firstName>John</firstName>
    <lastName>Smith</lastName>
    <mobile>0211223344</mobile>
  </person>
  <person staffDbId="-1" operation="add">
    <firstName>Mary</firstName>
    <lastName>Jane</lastName>
    <mobile>0244556677</mobile>
  </person>
</dailyTransaction>
```

complexType: dailyTransaction, person

simpleType: firstName, lastName, mobile

XSD: dailyTransaction example

```
<dailyTransaction date="24/02/2015">
  <person staffDbId="103" operation="update">
    ...
  </person>
  <person staffDbId="-1" operation="add">
    ...
  </person>
</dailyTransaction>
```

Start with the root element dailyTransaction:

```
<?xml version="1.0" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="dailyTransaction">
   <xsd:complexType>
   ...
   </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

XSD: dailyTransaction example

```
<dailyTransaction date="24/02/2015">
  <person staffDbId="103" operation="update">
    ...
  </person>
  <person staffDbId="-1" operation="add">
    ...
  </person>
</dailyTransaction>
```

The root element dailyTransaction contains a sequence of person elements and has attribute date

```
<xsd:element name="dailyTransaction">
   <xsd:complexType>
    <xsd:sequence>
      ...
    </xsd:sequence>
    <xsd:attribute name="date" type="xsd:string" />
   </xsd:complexType>
</xsd:element>
```

XSD: dailyTransaction example

```
<dailyTransaction date="24/02/2015">
  <person staffDbId="103" operation="update">
    ...
  </person>
  <person staffDbId="-1" operation="add">
    ...
  </person>
</dailyTransaction>
```

The root element dailyTransaction contains a sequence of person elements and has attribute date

```
<xsd:element name="dailyTransaction">
   <xsd:complexType>
    <xsd:sequence>
     <xsd:element name="person" minOccurs="0" maxOccurs="unbounded">
     ...
     </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="date" type="xsd:string" />
   </xsd:complexType>
</xsd:element>
```

XSD: dailyTransaction example

```
<person staffDbId="103" operation="update">
  <firstName>John</firstName>
  <lastName>Smith</lastName>
  <mobile>0211223344</mobile>
</person>
```

The element person contains:

- **elements: firstName, lastName, mobile**
- **attributes: staffDbId, operation**

```
<xsd:element name="person" minOccurs="0" maxOccurs="unbounded">
   <xsd:complexType>
    <xsd:sequence>
     ...
    </xsd:sequence>
    <xsd:attribute name="staffDbId" type="xsd:integer" />
    <xsd:attribute name="operation" type="xsd:string" />
   </xsd:complexType>
</xsd:element>
```

XSD: dailyTransaction example

```
<person staffDbId="103" operation="update">
  <firstName>John</firstName>
  <lastName>Smith</lastName>
  <mobile>0211223344</mobile>
</person>
```

The element person contains:

- **elements: firstName, lastName, mobile**
- **attributes: staffDbId, operation**

```
<xsd:element name="person" minOccurs="0" maxOccurs="unbounded">
   <xsd:complexType>
    <xsd:sequence>
       <xsd:element name="firstName" type="xsd:string"/>
       <xsd:element name="lastName" type="xsd:string"/>
       <xsd:element name="mobile" type="xsd:string"/>
    </xsd:sequence>
    <xsd:attribute name="staffDbId" type="xsd:integer" />
    <xsd:attribute name="operation" type="xsd:string" />
   </xsd:complexType>
</xsd:element>
```