

XML Data Management @ unibz - 2010/2011  
Chris Mair

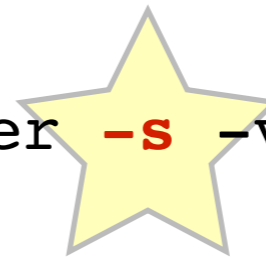
# W3C XML Schema

# Schemas compared

<b>Feature</b>	<b>DTD</b>	<b>W3C XML Schema</b>	<b>Relax NG</b>	<b>Schematron</b>
XML syntax	no	yes	yes	yes
compatible w. namespaces	no	yes	yes	yes
declares entities	yes	no	no	no
datatypes	no	yes	yes	yes
default attribute values	yes	yes	yes	no
notations	yes	yes	no	no
unordered content	no	yes	yes	yes
modular	yes	yes	yes	no
elem./attr. interchangeable	no	yes	yes	yes
specifies relation to doc.	yes	yes	no	no

# Example

- library.xml
- library\_1.xsd
- library\_2.xsd
- Source: “XML Schema” by Eric van der Vlist
- Validation: `xerces dom.Counter -s -v library.xml`



# Referencing schemas

- w/o namespaces

```
<library
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="library_1.xsd">
...
</library>
```

- w/ namespaces

```
<library
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://dyomedea.com/ns/library"
  xmlns:mkt="http://dyomedea.com/ns/library/mkt"
  xsi:schemaLocation="
    http://dyomedea.com/ns/library library.xsd
    http://dyomedea.com/ns/library/mkt marketing.xsd">
...
</library>
```

# Design: 2 approaches

- library\_1.xsd
- library\_2.xsd

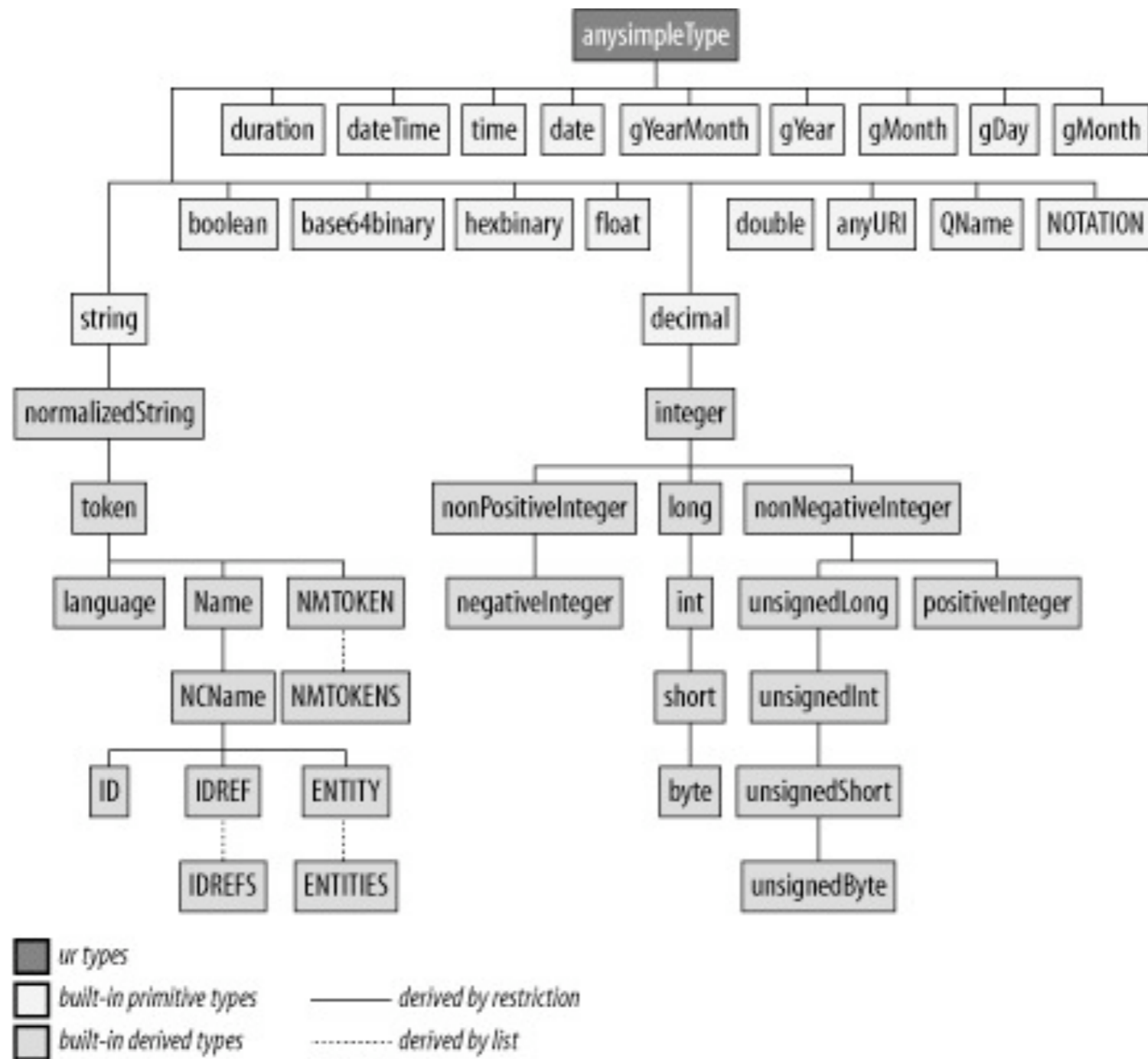
# More examples

- age.xml
- exam.xml
- names.xml
- unibzevent.xml
- unibzroom.xml / unibzroombad.xml
- and corresponding xsd files

# Data types 1/2

- `xs:string`
- `xs:decimal`
- `xs:integer`
- `xs:boolean` (true or false)
- `xs:date` (ISO-8601: e.g. *2009-10-21*)
- `xs:time` (ISO-8601: e.g. *10:30:00*)

# Data types 2/2



# Facets

- facets can restrict values to be in a given range (as seen in `age.xsd`):

```
<xs:element name="age">
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="130"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

- or be an element of an enumeration (`xs:enumeration`)

- or match a regular expression: `./.`

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

  <!-- this is a named type: it is not bound to any
        particular element, we can reuse it -->
  <xs:simpleType name="unibzroom_t">
    <!-- token is like string, but whitespace is ignored -->
    <xs:restriction base="xs:token">
      <!-- this is a regular expression:
            one letter between A and E followed by
            exactly 3 digits 0-9 -->
      <xs:pattern value="[A-E][0-9]{3}"/>
    </xs:restriction>
  </xs:simpleType>

  <!-- unibz is an element of type unibz_t -->
  <xs:element name="unibzroom" type="unibzroom_t"/>

</xs:schema>
```

```
$ grep E unibzroom.xml
  E411
$ xerces dom.Counter -s -v unibzroom.xml
unibzroom.xml: 258;9;0 ms (1 elems, 2 attrs, 0 spaces, 4 chars)
$
$ grep E unibzroom-bad.xml
  E41
$ xerces dom.Counter -s -v unibzroom-bad.xml
[Error] unibzroom-bad.xml:5:13: cvc-pattern-valid: Value 'E41' is not facet-valid with
respect to pattern '[A-E][0-9]{3}' for type 'unibzroom_t'.
[Error] unibzroom-bad.xml:5:13: cvc-type.3.1.3: The value 'E41' of element 'unibzroom'
is not valid.
unibzroom-bad.xml: 265;10;0 ms (1 elems, 2 attrs, 0 spaces, 3 chars)
$
```

# Complex types

- elements that contain other elements, attributes or character data can be defined using complex types
- elements of complex types can be constraint to contain all (`xs:all`), just one (`xs:choice`) or a repetition (`xs:sequence`) of elements from a given list
- the attribute `mixed="true"` allows character data between elements

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs='http://www.w3.org/2001/XMLSchema'>

  <!-- element "name" must contain two elements,
        "first" and "last", both of type string -->
  <xs:element name="name">
    <xs:complexType>
      <xs:all>
        <xs:element name="first" type="xs:string"/>
        <xs:element name="last" type="xs:string"/>
      </xs:all>
    </xs:complexType>
  </xs:element>

  <!-- element "names" must contain a list of one
        or more elements "name" -->
  <xs:element name="names">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="name" minOccurs='1' maxOccurs='unbounded' />
      </xs:sequence>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

```
<?xml version="1.0"?>
<names xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation='names.xsd'>
  <name>
    <first>John</first>
    <last>von Neumann</last>
  </name>
  <name>
    <last>Turing</last>
    <first>Alan</first>
  </name>
</names>
```

# Attributes

- Attributes are defined the same way as elements are, e.g.:

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

  <xs:element name="exam">
    <xs:complexType mixed="true">
      <xs:attribute name="title" type="xs:string" use="required"/>
      <xs:attribute name="mark" type="xs:integer" use="required"/>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

# The empty element

- there is no built-in definition for an empty element, you need to roll your own:

```
<xs:complexType name="empty_t" />
```

# Comparison to RDBMs

compare this to age.xsd!

```
Welcome to psql 8.3.6, the PostgreSQL interactive terminal.
```

```
chris=# create table person (age int not null,  
constraint chk_age check(age >= 0 and age <= 130));
```

```
CREATE TABLE
```

```
chris=#
```

```
chris=# insert into person values (17);
```

```
INSERT 0 1
```

```
chris=# insert into person values (0);
```

```
INSERT 0 1
```

```
chris=# insert into person values (100);
```

```
INSERT 0 1
```

```
chris=# insert into person values (200);
```

```
ERROR: new row for relation "person" violates  
check constraint "chk_age"
```

# A cool Postgres Feature

```
chris=# select table_to_xmlschema('person', false, false, '');
```

```
table_to_xmlschema
```

---

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

  <xsd:simpleType name="INTEGER">
    <xsd:restriction base="xsd:int">
      <xsd:maxInclusive value="2147483647"/>
      <xsd:minInclusive value="-2147483648"/>
    </xsd:restriction>
  </xsd:simpleType>

  <xsd:complexType name="RowType.chris.public.person">
    <xsd:sequence>
      <xsd:element name="age" type="INTEGER" minOccurs="0"/></xsd:element>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="TableType.chris.public.person">
    <xsd:sequence>
      <xsd:element name="row" type="RowType.chris.public.person" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:element name="person" type="TableType.chris.public.person"/>

</xsd:schema>
(1 row)
```

# Real World Example

- <http://www.opentravel.org/>
- OTA 2010A
- <http://adriatic.pilotfish-net.com/ota-modelviewer/>