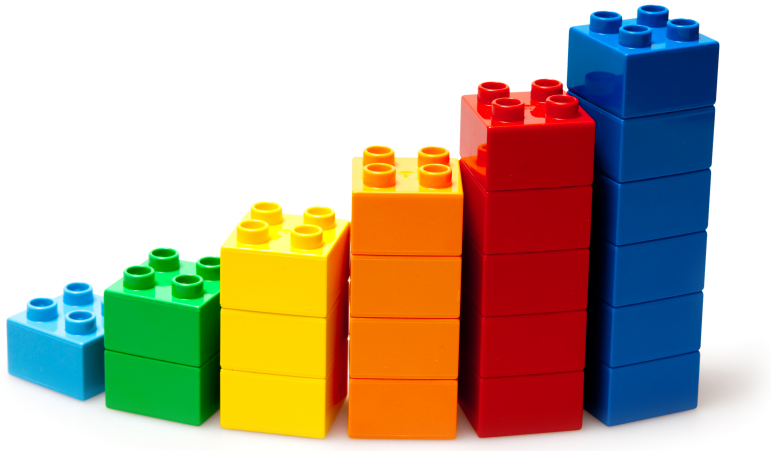
Introduction to Database Systems: CS312
XML and Databases Schemas

Oliver Bonham-Carter

13 March 2019

Schema



- The purpose of an XML Schema is to define the legal building blocks of an XML document:
  - The elements and attributes that can appear in a document
  - The number of (and order of) child elements
  - Data types for elements and attributes
  - Default and fixed values for elements and attributes

Data Types and Restrictions

- XML Schema supports data types
- Greatest strength of XML schema is the support for data types
  - Describe allowable document content
  - Validate the correctness of data
  - Define data facets (restrictions on data)
  - Define data patterns (data formats)
  - Convert data between different data types

Structural Requirements

- XML Schema: Allows for
  - Basic structural requirements for data
  - Usage of built-in types, such as *string*, *integer*, *decimal* *date*, and *boolean* variables.
  - Content-specific descriptors for XML data.
  - User-defined types; these may be simple types with added restrictions, or complex types constructed using constructors such as *complexType* and *sequence*.

Basic File Format


```
<?xml version="1.0"?>
<note xmlns="https://www.w3schools.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https://www.w3schools.com note.xsd">
     <to>James Bond</to>
     <from>M</from>
     <heading>Reminder</heading>
     <body>The Aston Martin needs oil work!</body>
</note>
```

NameSpaces

- *XML Namespaces* provide a method to avoid element name conflicts
- `xmlns:xs="http://www.w3.org/2001/XMLSchema"`
  - Namespace declaration: elements and data types are to be prefixed with `xs:`
- `targetNamespace="https://www.w3schools.com"`
  - Namespace declaration: defined elements come from w3schools namespace, `xmlns` is default namespace.
- `elementFormDefault="qualified"` means all elements in this xml document must be namespace qualified

Conflicts in Environments?

- In XML, element names are defined by the developer and conflicts arise when combining XML documents from different XML applications
- Both contain a <table> element, but each element has different contents and meanings

```
<table>
  <tr>
    <td>Apples</td>
    <td>Bananas</td>
  </tr>
</table>
```

```
<table>
  <name>African Coffee Table</name>
  <width>80</width>
  <length>120</length>
</table>
```

Discerning Tables

Try this...

- Use name prefix to differentiate one table from another. This is code contained in a XML document.

```
<h:table>
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>

<f:table>
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
```

Prefixes

What that did...

- Namespaces defined by an `xmlns` attribute in element's the start tag
- The `xmins` gives the first and second tables a qualified `h:` or `f` prefix, respectively, for differentiation

```
<root>
<h:table xmlns:h="http://www.w3.org/TR/html4/">
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>

<f:table xmlns:f="https://www.w3schools.com/furniture">
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
```

Full XML Code with Differentiating Tables

- The namespace declaration has the following syntax: `xmlns:prefix="URI"`

```
<root xmlns:h="http://www.w3.org/TR/html4/"
xmlns:f="https://www.w3schools.com/furniture">
<h:table>
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>
<f:table>
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
</root>
```

Schemas Support

No schema information...

```
<person>
  <firstname>John</firstname>
  <lastname>Smith</lastname>
</person>
```

```
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

- <xs:sequence> the elements ("firstname" and "lastname") must appear in this order inside the "person" element.

Schemas Support


```
<xs:element name="person" type="persontype"/>

<xs:complexType name="persontype">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

Attributes for the elements

- Give the `complexType` element the name `persontype`.
- We give the `person` element a type attribute, `complexType`
- With this method, several elements can refer to the same complex type with same code.

Declaring an Element

- Elements are declared using: `<xs:element>` tags in the schema file
  - `<xs:element name="x" type="y" />`
  - `<xs:element name="Birthday" type="xs:date" />`
- Can be declared as having a simple or complex type
- Elements can have mixed, empty or element context
- Elements can be given a minimum or maximum number of times that they are allowed occur
- Elements restricted to having specific values (similar to *integrity constraints*)

Some Element Declarations

Built-in simple types

Simple	Description	Example
xs:string	sequence of characters	"Hello XML"
xs:boolean	a boolean value	true, false, 0 or 1
xs:decimal	a float point number	-13.3
xs:integer	a number with no decimal	13
xs:positiveInteger	an int greater than 0	4
xs:negativeInteger	an int less than 0	-15
xs:date	calendar date: CCYY-MM-DD	2019-03-13
xs:gMonth	calendar month: - -MM- -	–11– (Nov.)
xs:gDay	calendar day: - - -DD	- - -26
xs:anyURI	URL $\subseteq$ URI	`www.allegheny.edu`

Type Examples

- <xs:element name = "Name" type ="xs:string" />
- <xs:element name="startdate" type="xs:date" />
- <xs:element name = "friend" type ="xs:string" minoccurs="1" maxoccurs="unbounded" />
  - A constraints for a type:
    - We may define how many times that the element can appear in an block.
    - *minoccurs*: indicates that the `friend` element can occur a minimum of once
    - *maxoccurs*: indicated an unlimited number of occurrences in an element's environment.

Schemas

- XML Schemas are written in XML
- XML Schemas are extensible to additions
- XML Schemas support data types
- XML Schemas support namespaces

```
<xs:element name="note">
<xs:complexType>
  <xs:sequence>
    <xs:element name="to" type="xs:string"/>
    <xs:element name="from" type="xs:string"/>
    <xs:element name="heading" type="xs:string"/>
    <xs:element name="body" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
</xs:element>
```

Schemas

The Schema above is interpreted like this:

- `<xs:element name="`**note**`">` defines note element
- `<xs:complexType>` the **note** is a complex type
- `<xs:sequence>` complex type is sequence of elements
- `<xs:element name="to" type="`**xs:string**`">`: The element "to" is of type string (text)
- `<xs:element name="from" type="`**xs:string**`">` The element "from" is of type string
- `<xs:element name="heading" type="`**xs:string**`">`: The element "heading" is of type string
- `<xs:element name="body" type="`**xs:string**`">`: The element "body" is of type string

File: noteData.xml

Contains the schema

Working XML code

```
<?xml version="1.0"?>
<note
   xmlns="http://www.w3schools.com"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="note_schema.xsd">
       <to>Ken</to>
       <from>Barbie</from>
       <heading>Reminder</heading>
       <body>See you at lunch!</body>
</note>
```

File: noteSchema.xsd

Contains the schema definitions for note_data.xml


```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.w3schools.com"
xmlns="http://www.w3schools.com"
elementFormDefault="qualified">

<xs:element name="note">
<xs:complexType>
  <xs:sequence>
<xs:element name="to" type="xs:string"/>
<xs:element name="from" type="xs:string"/>
<xs:element name="heading" type="xs:string"/>
<xs:element name="body" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

Are Elements Correctly "Configured"?

Use `xmllint`

`xmllint`

- A element and general syntax checker of schema definition files.
- For determining whether elements are correctly configured (programmed) between the xml (data containing) and xsd (definition) files
- The XML C parser and toolkit of Gnome::*libxml*
- `http://xmlsoft.org/intro.html`

Information about *xmllint*

"Programming with libxml2 is like the thrilling embrace of an exotic stranger." Mark Pilgrim

Libxml2 is the XML C parser and toolkit developed for the Gnome project (but usable outside of the Gnome platform), it is free software available under the MIT License. XML itself is a metalanguage to design markup languages, i.e. text language where semantic and structure are added to the content using extra "markup" information enclosed between angle brackets. HTML is the most well-known markup language. Though the library is written in C a variety of language bindings make it available in other environments.

Libxml2 is known to be very portable, the library should build and work without serious troubles on a variety of systems (Linux, Unix, Windows, CygWin, MacOS, MacOS X, RISC Os, OS/2, VMS, QNX, MVS, VxWorks, ...)

Libxml2 implements a number of existing standards related to markup languages:

- the XML standard: http://www.w3.org/TR/REC-xml
- Namespaces in XML: http://www.w3.org/TR/REC-xml-names/
- XML Base: http://www.w3.org/TR/xmlbase/
- RFC 2396 : Uniform Resource Identifiers http://www.ietf.org/rfc/rfc2396.txt
- XML Path Language (XPath) 1.0: http://www.w3.org/TR/xpath
- HTML4 parser: http://www.w3.org/TR/html401/
- XML Pointer Language (XPointer) Version 1.0: http://www.w3.org/TR/xptr
- XML Inclusions (XInclude) Version 1.0: http://www.w3.org/TR/xinclude/
- ISO-8859-x encodings, as well as rfc2044 [UTF-8] and rfc2781 [UTF-16] Unicode encodings,

xmllint Terminal Command

How do you know that your schema and data work together?

- We will verify (*validate*) that both files are correctly configured.
- Any errors where the elements are not correctly used will result in an error

```
xmllint -schema noteSchema.xsd --noout noteData.xml
```

If your schema and data are correct:

```
noteData.xml validates
```

If your schema and data are NOT correct:

```
noteData_i.xml:7: element too: Schemas validity error :
Element '{http://www.w3schools.com}too':
This element is not expected.
 Expected is ( {http://www.w3schools.com}to ).
noteData_i.xml fails to validate
```

Consider This...

Copy the *notes* files and update with the following ...

- Create a `lunchAddress` field and add the following address

Address for Lunch Place

- **The Lunch Hack**, 6001 Dodge Street, Omaha, NE 68182

- Verify your work with `xmllint`
- Can you add, YET MORE, fields for `Integers` and a `Date`?
- Be sure to verify your work again