Semi Structured Data

3rd year computer science (license)

2025/2026

Chapter 3
XML CORE

Chapter plan

Introduction to XML

Basic XML Structure

Namespaces

XML Schemas

Introduction to XML

3.1.1 What is XML?

XML (eXtensible Markup Language) is a **metalanguage** - a language used to create other markup languages. Developed by the W3C (World Wide Web Consortium) in 1996, XML was designed to store and transport data with both human readability and machine processing in mind.

Introduction to XML

Key Characteristics:

- **Extensible:** Users can create their own tags
- **Platform-independent:** Works across all operating systems
- **Self-descriptive:** Tags convey meaning about the data
- **Text-based:** Readable by humans and machines
- **Unicode-compliant:** Supports international characters

Introduction to XML

3.1.2 XML vs HTML: A Critical Distinction

Feature	XML	HTML
Purpose	Data description and transport	Data presentation
Tags	User-defined, extensible	Predefined, fixed set
Structure	Must be well-formed	Lenient, forgiving
Case sensitivity	Yes (case-sensitive)	No (case-insensitive)
Whitespace	Preserves all whitespace	Collapses multiple spaces
Error handling	Errors stop processing	Browser attempts to render anyway

Introduction to XML

3.1.3 The XML Family

XML is not just a language but an ecosystem including:

- **XPath:** For navigating XML documents
- **XSLT:** For transforming XML
- **XML Schema:** For defining structure
- **XQuery:** For querying XML data
- **DOM/SAX:** For programmatic processing

Introduction to XML

3.1.4 Real-World Applications of XML

- **Configuration Files:** Maven's pom.xml, Android's AndroidManifest.xml
- **Data Exchange:** SOAP web services, RSS/Atom feeds
- **Office Documents:** Microsoft Office (DOCX, XLSX), OpenDocument format
- **Industry Standards:** MathML (mathematics), SVG (graphics), HL7 (healthcare)

Basic XML Structure

The prolog is optional but recommended. It appears at the very beginning of an XML document:

- <?xml version="1.0" encoding="UTF-8" standalone="yes"?>

Components:

- version: Usually "1.0" or "1.1"
- encoding: Character encoding (UTF-8, UTF-16, ISO-8859-1)
- standalone: Whether the document depends on external markup declarations

Basic XML Structure

3.2.2 XML Elements: The Building Blocks

Elements are the primary containers for data:

<elementName>Content goes here</elementName>

Rules for Element Names:

- Can contain letters, digits, hyphens, underscores, periods
- Cannot start with a digit or punctuation (except underscore)
- Cannot start with "xml" (any case variation)
- Cannot contain spaces

Basic XML Structure

Examples:

```
<!-- Valid element names -->
<book>
<book_title>
<bookTitle>
<book-title>
<book2>
<_book>
```

```
<!-- Invalid element names -->
<2book>     <!-- Starts with digit -->
<book title> <!-- Contains space -->
<xml-book>   <!-- Starts with xml -->
<book!>      <!-- Contains invalid characte
r -->
```

Basic XML Structure

3.2.3 Element Content Types

Elements can contain different types of content:

- **Element Content:** Nested elements only

```
<address>
    <street>123 Main St</street>
    <city>Boston</city>
</address>
```

Basic XML Structure

Simple Content: Text only

```
<price currency="USD">29.99</price>
```

Mixed Content: Both text and elements

```
<description>
   This book is <emphasis>highly recommended</emphasis> for beginners.
</description>
```

Basic XML Structure

Empty Content: No content, can be self-closing

<line-break/>

<image src="photo.jpg" />

Basic XML Structure

3.2.4 Attributes: Adding Metadata

Attributes provide additional information about elements:

```
<book isbn="978-3-16-148410-0" language="en" edition="2">
  <title>XML Fundamentals</title>
</book>
```

Attribute Rules:

- Each attribute must have a unique name within an element
- Attribute values must be quoted (single or double quotes)
- Cannot contain <, &, or > in attribute values (use entities)

Basic XML Structure

When to Use Elements vs Attributes:

Use Elements When	Use Attributes When
Data is hierarchical	Data is metadata about the element
Data might have complex structure	Data is simple, atomic values
Data needs to be extensible	Data is identification information
Data contains long text	Data is enumerations (IDs, codes)

Basic XML Structure

3.2.5 Well-Formedness Rules

A well-formed XML document must satisfy these rules:

- **One Root Element:** The document must have exactly one root element

```
<!-- CORRECT -->
<library>
   <book>...</book>
   <book>...</book>
</library>

<!-- INCORRECT - multiple roots -->
<book>...</book>
<book>...</book>
```

Basic XML Structure

Proper Nesting: Elements must be properly nested, not overlapping

<!-- CORRECT -->

<bold><italic>Important text</italic></bold>

<!-- INCORRECT - overlapping -->

<bold><italic>Important text</bold></italic>

Case Sensitivity: Start and end tags must match exactly

<!-- CORRECT -->

<BookTitle>XML Guide</BookTitle>

<!-- INCORRECT - case mismatch -->

<BookTitle>XML Guide</booktitle>

Special Character Handling: Use entities for reserved characters

<!-- CORRECT -->

<equation>5 < 10 && 10 > 5</equation>

<!-- INCORRECT - raw special characters -->

<equation>5 < 10 && 10 > 5</equation>

Basic XML Structure

3.2.6 XML Entities

XML provides built-in entities for special characters:

Entity	Character	Description
&lt;	<	Less than
&gt;	>	Greater than
&amp;	&	Ampersand
&apos;	'	Apostrophe
&quot;	"	Quotation mark

Basic XML Structure

User-defined entities can also be declared in DTDs:

```
<!DOCTYPE chapter [
   <!ENTITY copyright "© 2024, All Rights Reserved">
]>
<chapter>
   <para>&copyright;</para>
</chapter>
```

Basic XML Structure

3.2.7 Comments and Processing Instructions

Comments: For human-readable notes

<!-- This is a comment -->

<!-- Comments can span

multiple lines -->

<!-- WARNING: Do not modify this section -->

Processing Instructions: For application-specific instructions

<?xml-stylesheet type="text/xsl" href="style.xsl"?>

<?php echo "Dynamic content"; ?>

Basic XML Structure

3.2.8 CDATA Sections

CDATA sections tell the parser to treat content as character data, not markup:

```
<code-snippet>
   <![CDATA[
      if (x < y && y > z) {
         System.out.println("x < y && y > z");
      }
   ]]>
</code-snippet>
```

CDATA Rules:

- Cannot contain the string "]]>"
- Useful for embedding code or XML examples within XML
- Preserves whitespace exactly as written

Namespaces

3.3.1 The Problem Namespaces Solve

Consider merging XML from different sources:

```
<!-- From inventory system -->
<table>
   <material>Oak</material>
   <legs>4</legs>
</table>

<!-- From database schema -->
<table>
   <column>Name</column>
   <column>Age</column>
</table>
```

When combined, which <table> means furniture and which means database table? Namespaces resolve this ambiguity

Namespaces

3.3.2 Namespace Syntax

Declaration: xmlns:prefix="URI"

```
<root xmlns:inv="http://www.example.com/inventory"
      xmlns:db="http://www.example.com/database">

   <inv:table>
      <inv:material>Oak</inv:material>
   </inv:table>

   <db:table>
      <db:column>Name</db:column>
   </db:table>
</root>
```

Namespaces

3.3.3 Default Namespace

A namespace without a prefix applies to all unprefixed elements:

```
<library xmlns="http://www.example.com/library"
        xmlns:auth="http://www.example.com/author">

   <!-- 'book' is in library namespace -->
   <book>
      <title>XML Guide</title>
      <!-- 'author' element is also in library namespace -->
      <author>
         <!-- 'name' is in author namespace -->
         <auth:name>John Doe</auth:name>
      </author>
   </book>
</library>
```

Namespaces

3.3.4 Namespace Scope

Namespaces apply to the declaring element and its descendants:

```
<root>
   <child xmlns:ns1="http://example.com/ns1">
      <ns1:element>In scope</ns1:element>
      <grandchild xmlns:ns2="http://example.com/ns2">
         <ns2:element>In scope</ns2:element>
         <ns1:element>Still in scope</ns1:element>
      </grandchild>
   </child>
   <!-- ns1: not in scope here -->
</root>
```

Namespaces

3.3.5 Attributes and Namespaces

Attributes don't inherit the default namespace and need explicit prefixes:

```
<book xmlns="http://example.com/books"
      xmlns:xlink="http://www.w3.org/1999/xlink"
      xlink:type="simple"
      xlink:href="http://example.com/book1">

    <title>XML Namespaces</title>
    <!-- 'lang' attribute is in no namespace -->
    <description lang="en">A comprehensive guide</description>
</book>
```

XML Validation

A "well formed" XML document is not the same as a "valid" XML document.

A "valid" XML document must be well formed. In addition, it must conform to a document type definition.

There are two different document type definitions that can be used with XML:

- DTD - The original Document Type Definition
- XML Schema - An XML-based alternative to DTD

A document type definition defines the rules and the legal elements and attributes for an XML document.

Document Type Definition (DTD)

Document Type Definition (DTD) is a set of markup declarations that define the structure, elements, and attributes of an XML document. It's the original schema language for XML, inherited from SGML.

Key Characteristics:

- **Syntax:** Non-XML syntax (though there is an XML syntax called XML Schema)
- **Purpose:** Defines the legal building blocks of an XML document
- **Validation:** Ensures XML documents follow predefined rules
- **Location:** Can be internal (within XML) or external (separate .dtd file)

Document Type Definition (DTD)

A DTD can be defined in two ways:

```
<!DOCTYPE library [
  <!ELEMENT library (book+)>
  <!ELEMENT book (title, author, year)>
  <!ELEMENT title (#PCDATA)>
  <!ELEMENT author (#PCDATA)>
  <!ELEMENT year (#PCDATA)>
]>Internal DTDDefined inside the XML document.
```

Document Type Definition (DTD)

External DTD
Defined in a separate file (library.dtd):

```
<!ELEMENT library (book+)>
<!ELEMENT book (title, author, year)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT year (#PCDATA)>
```

Linked in XML:

```
<!DOCTYPE library SYSTEM "library.dtd">
```

Document Type Definition (DTD)

1. Element Declarations (<!ELEMENT>)

1.1 Basic Element Declaration Syntax

<!ELEMENT elementName contentModel>

1.2 Element Content Models

EMPTY Elements

Elements with no content (self-closing tags):

<!ELEMENT line-break EMPTY>

<!ELEMENT image EMPTY>

<!ELEMENT hr EMPTY>

XML Instance:

```
<line-break/>
<image src="photo.jpg"/>
<hr/>
```

Document Type Definition (DTD)

ANY Elements

Can contain any content (rarely used, poor for validation):

<!ELEMENT note ANY>

XML Instance (all valid):

<note>Just text</note>

<note><bold>Mixed</bold> content</note>

<note></note>

Document Type Definition (DTD)

#PCDATA Elements

Parsed Character Data - text-only content:

<!ELEMENT title (#PCDATA)>

<!ELEMENT price (#PCDATA)>

<!ELEMENT description (#PCDATA)>

XML Instance:

<title>XML Fundamentals</title>

<price>49.99</price>

<description>This is a text-only description</description>

Document Type Definition (DTD)

Element Content (Children Only)

Elements that contain only other elements:

<!ELEMENT address (street, city, zip)>

<!ELEMENT book (title, author, price)>

XML Instance:

```
<address>
   <street>123 Main St</street>
   <city>Boston</city>
   <zip>02101</zip>
</address>
```

Document Type Definition (DTD)

Mixed Content

Elements that contain both text and child elements:

<!ELEMENT paragraph (#PCDATA | bold | italic)*>

XML Instance:

<paragraph>

This is <bold>important</bold> text with <italic>emphasis</italic>.

</paragraph>

Document Type Definition (DTD)

2. Cardinality Operators

Operators that specify how many times an element can appear:

Operator	Meaning	Example
(none)	Exactly once	(title)
?	Zero or one (optional)	(subtitle?)
+	One or more	(paragraph+)
*	Zero or more	(comment*)

Document Type Definition (DTD)

2.1 Detailed Examples with Cardinality

```
<!ELEMENT book (isbn, title, subtitle?, author+, category*, price)>

<!-- Explanation:
   - isbn: exactly one (required)
   - title: exactly one (required)
   - subtitle: zero or one (optional)
   - author: one or more (at least one author required)
   - category: zero or more (any number of categories, including none)
   - price: exactly one (required)
-->
```

Document Type Definition (DTD)

Valid XML Instances:

```
<!-- Minimal valid book -->
<book>
    <isbn>978-3-16-148410-0</isbn>
    <title>XML Basics</title>
    <author>John Doe</author>
    <price>29.99</price>
</book>
```

```
<!-- Complete book with all optional elements
<book>
    <isbn>978-3-16-148410-0</isbn>
    <title>Advanced XML</title>
    <subtitle>Comprehensive Guide</subtitle>
    <author>John Doe</author>
    <author>Jane Smith</author>
    <category>Programming</category>
    <category>XML</category>
    <category>Reference</category>
    <price>49.99</price>
</book>
```

Document Type Definition (DTD)

3.1 Sequence (,)

Elements must appear in the specified order:

```
<!ELEMENT person (firstName, lastName, age, email?)>
<person>
  <firstName>John</firstName>
   <lastName>Doe</lastName>
   <age>30</age>
   <email>john@example.com</email>
</person>
<!-- INVALID: Wrong order -->
<person>
   <lastName>Doe</lastName>  <!-- Error: lastName before firstName -->
   <firstName>John</firstName>
   <age>30</age>
</person>
```

Document Type Definition (DTD)

3.2 Choice (|)

One (and only one) of the elements must appear:

```
<!ELEMENT contact (email | phone | address)>
<!-- Valid: email chosen -->
<contact>
   <email>user@example.com</email>
</contact>
<!-- Valid: phone chosen -->
<contact>
   <phone>555-0123</phone>
</contact>
```

```
<!-- INVALID: Both chosen -->
<contact>
   <email>user@example.com</email>
   <phone>555-0123</phone>  <!-- Error: Can't have both -->
</contact>
<!-- INVALID: None chosen -->
<contact>
</contact>  <!-- Error: Must have one of the options -->
```

Document Type Definition (DTD)

3.3 Complex Combinations

Combining sequence and choice with cardinality:

```
<!ELEMENT order (customer, (item+ | service+), payment, shipping?)>
<!-- Order with items -->
<order>
   <customer>Acme Corp</customer>
   <item>Product A</item>
   <item>Product B</item>
   <payment>Credit Card</payment>
</order>
```

```
<!-- Order with services -->
<order>
   <customer>Acme Corp</customer>
   <service>Consulting</service>
   <service>Training</service>
   <payment>Invoice</payment>
   <shipping>Express</shipping>
</order>
```

```
<!-- INVALID: Can't mix items and services -->
<order>
   <customer>Acme Corp</customer>
   <item>Product A</item>
   <service>Consulting</service>  <!-- Error: Can't mix types -->
   <payment>Credit Card</payment>
</order>
```

Document Type Definition (DTD)

3.4 Nested Groups

Parentheses can group elements for complex rules:

<!ELEMENT book (title, (author | editor)+, (chapter | appendix)*, index?)>

<!ELEMENT chapter (title, (paragraph | figure | table)+, (section*))>

<!ELEMENT section (title, (paragraph | note | warning)*, (subsection*))
>

Document Type Definition (DTD)

4. Attribute Declarations (<!ATTLIST>)

4.1 Basic Attribute Declaration Syntax

```
<!ATTLIST elementName
   attributeName attributeType attributeDefault
   attributeName attributeType attributeDefault
   ...
>
```

Document Type Definition (DTD)

4.2 Attribute Types

CDATA (Character Data) Simple text values:

```
<!ELEMENT book EMPTY>
<!ATTLIST book
   isbn CDATA #REQUIRED
   title CDATA #REQUIRED
   language CDATA "English"
   edition CDATA #IMPLIED >
```

XML Instance:

```
<book isbn="978-3-16-148410-0"
     title="XML Guide"
     language="French"
     edition="2"/>
<book isbn="978-0-12-345678-9"
     title="Another Book"
     language="English"/>  <!-- edition is implied (optional) -->
```

Document Type Definition (DTD)

ID (Unique Identifier) Must be unique within the XML document:

```
<!ELEMENT person EMPTY>
<!ATTLIST person
   personID ID #REQUIRED
   name CDATA #REQUIRED >

<!ELEMENT company EMPTY>
<!ATTLIST company
   companyID ID #REQUIRED
   name CDATA #REQUIRED >
```

XML Instance:

```
<person personID="p123" name="John Doe"/>
<person personID="p456" name="Jane Smith"/>
<company companyID="c789" name="Acme Inc."/>
<!-- INVALID: Duplicate ID -->
<person personID="p123" name="Another John"/>  <!-- Error: ID p123 already used -->
```

Document Type Definition (DTD)

ID Rules:

- Must start with a letter or underscore
- Cannot start with a number
- Must be unique across ALL ID attributes in the document
- Cannot contain spaces

Document Type Definition (DTD)

IDREF (ID Reference) References an existing ID elsewhere in the document:

```
<!ELEMENT employee EMPTY>
<!ATTLIST employee
    empID ID #REQUIRED
    name CDATA #REQUIRED
    managerID IDREF #IMPLIED >
 <!ELEMENT department EMPTY>
<!ATTLIST department
    deptID ID #REQUIRED
    name CDATA #REQUIRED
    headID IDREF #REQUIRED >
```

XML Instance:

```
<employee empID="e001" name="John Doe"/>
<employee empID="e002" name="Jane Smith" managerID="e001"/>
<department deptID="d001" name="IT" headID="e002"/>
<employee empID="e003" name="Bob Johnson" managerID="e001"/>
<!-- INVALID: References non-existent ID -->
<employee empID="e004" name="Alice Brown" managerID="e999"/>
<!-- Error: e999 doesn't exist -->
```

Document Type Definition (DTD)

IDREFS (Multiple ID References) Space-separated list of ID references:

```
<!ELEMENT project EMPTY>
<!ATTLIST project
    projID ID #REQUIRED
    name CDATA #REQUIRED
    teamMembers IDREFS #REQUIRED>
<!ELEMENT employee EMPTY>
<!ATTLIST employee
    empID ID #REQUIRED
    name CDATA #REQUIRED >
```

XML Instance:

```
<employee empID="e001" name="John"/>
<employee empID="e002" name="Jane"/>
<employee empID="e003" name="Bob"/>
<project projID="p001"
        name="XML Project"
        teamMembers="e001 e002 e003"/>
```

Document Type Definition (DTD)

ENUMERATION (Fixed List of Values) Must be one of the specified values:

```
<!ELEMENT product EMPTY>
<!ATTLIST product
   name CDATA #REQUIRED
   size (small | medium | large | x-large) "medium"
   color (red | green | blue | black | white) #REQUIRED
   status (active | discontinued | pending) "active«  >
```

XML Instance:

```
<product name="T-Shirt" size="large" color="blue" status="active"/>
<product name="Mug" color="white" status="active"/>
<product name="Hat" size="small" color="red" status="discontinued"/>
<!-- INVALID: Not in enumeration -->
<product name="Pants" size="extra-large" color="blue"/> <!-- Error: extra-large not allowed -->
<product name="Socks" color="purple"/> <!-- Error: purple not in color enumeration -->
```

Document Type Definition (DTD)

4.3 Attribute Defaults

#REQUIRED

Attribute must always be present:

```
<!ELEMENT user EMPTY>
<!ATTLIST user
   username CDATA #REQUIRED
   password CDATA #REQUIRED >
```

XML Instance:

```
<user username="john" password="secret"/>
<!-- INVALID: Missing required attributes -->
<user username="john"/>  <!-- Error: password is required -->
<user password="secret"/>  <!-- Error: username is required -->
```

Document Type Definition (DTD)

#IMPLIED

Attribute is optional:

```
<!ELEMENT article EMPTY>
<!ATTLIST article
   title CDATA #REQUIRED
   author CDATA #IMPLIED
   date CDATA #IMPLIED >
```

XML Instance:

```
<article title="XML Basics" author="John" date="2024-01-15"/>
<article title="DTD Guide"/> <!-- Valid: author and date are implied -->
<article title="XPath Tutorial" author="Jane"/> <!-- Valid: date is implied -->
```

Document Type Definition (DTD)

Default Value

Attribute has a default value if not specified:

```
<!ELEMENT book EMPTY>
<!ATTLIST book
   title CDATA #REQUIRED
   language CDATA "English"
   format CDATA "hardcover"
   pages CDATA #IMPLIED >
```

XML Instance:

```
<!-- language="English", format="hardcover" are defaulted -->
<book title="XML Guide"/>
<!-- Override defaults -->
<book title="French XML" language="French" format="paperback" pages="300"/>
<!-- Partial override -->
<book title="Spanish Guide" language="Spanish" pages="250"/>  <!-- format defaults to "hardcover" -->
```

Document Type Definition (DTD)

#FIXED

Attribute has a fixed, unchangeable value:

```
<!ELEMENT document EMPTY>
<!ATTLIST document
   version CDATA #FIXED "1.0"
   encoding CDATA #FIXED "UTF-8"
   title CDATA #REQUIRED >
```

XML Instance:

```
<!-- Valid: version and encoding are fixed -->
<document title="My Doc"/>
<!-- Also valid: must match fixed value -->
<document title="My Doc" version="1.0" encoding="UTF-8"/>
<!-- INVALID: Can't change fixed value -->
<document title="My Doc" version="2.0"/> <!-- Error: version must be "1.0" -->
<document title="My Doc" encoding="ASCII"/> <!-- Error: encoding must be "UTF-8" -->
```

XML Schemas

3.4.1 Why Schemas?

Schemas define the vocabulary, structure, and constraints for XML documents:

- **Problems Solved by Schemas:**
- **Validation:** Ensure documents follow rules
- **Documentation:** Formal specification of document structure
- **Data Typing:** Specify data types (integer, date, etc.)
- **Constraint Definition:** Define required/optional elements, repetitions
- **Interoperability:** Ensure different systems understand the same format

XML Schemas

3.4.2 XML Schema vs DTD

Feature	XML Schema (XSD)	DTD
Syntax	XML-based	Non-XML
Data Types	Rich type system (40+ types)	Limited (only PCDATA, CDATA)
Namespaces	Full support	Limited
Extensibility	Can derive new types	Not extensible
Cardinality	minOccurs/maxOccurs	*, +, ? only
Primary use	Modern XML validation	Legacy systems

XML Schemas

3.4.3 Schema Document Structure

A basic XML Schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.example.com/books"
    xmlns="http://www.example.com/books"
    elementFormDefault="qualified">
    <!-- Schema definitions go here -->
</xs:schema>
```

Key Attributes:

- targetNamespace: The namespace for elements defined in this schema
- elementFormDefault: Whether locally declared elements must be qualified

XML Schemas

3.4.4 Simple Types

Simple types contain only text content, no elements or attributes:

Built-in Types:

```
<xs:element name="age" type="xs:integer"/>
<xs:element name="price" type="xs:decimal"/>
<xs:element name="published" type="xs:date"/>
<xs:element name="email" type="xs:string"/>
<xs:element name="inStock" type="xs:boolean"/>
```

XML Schemas

Restricting Simple Types (Facets):

```
<xs:element name="age">
   <xs:simpleType>
      <xs:restriction base="xs:integer">
         <xs:minInclusive value="0"/>
         <xs:maxInclusive value="150"/>
      </xs:restriction>
   </xs:simpleType>
</xs:element>
```

```
<xs:element name="password">
   <xs:simpleType>
      <xs:restriction base="xs:string">
         <xs:minLength value="8"/>
         <xs:maxLength value="20"/>
         <xs:pattern value="[A-Za-z0-9]+"/>
      </xs:restriction>
   </xs:simpleType>
</xs:element>
```

XML Schemas

Common Facets:

Facet	Description	Example
minLength/maxLength	String length constraints	Password length
pattern	Regular expression pattern	Email format
minInclusive/maxInclusive	Numeric range	Age range
enumeration	List of allowed values	Colors, statuses
totalDigits/fractionDigits	Decimal precision	Currency values

XML Schemas

3.4.5 Complex Types

Complex types can contain elements and attributes:

Sequence (ordered elements):

```
<xs:element name="book">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="title" type="xs:string"/>
      <xs:element name="author" type="xs:string"/>
      <xs:element name="price" type="xs:decimal"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

XML Schemas

Choice (one of several options):

```
<xs:element name="contact">
  <xs:complexType>
    <xs:choice>
      <xs:element name="email" type="xs:string"/>
      <xs:element name="phone" type="xs:string"/>
      <xs:element name="address" type="addressType"/>
    </xs:choice>
  </xs:complexType>
</xs:element>
```

XML Schemas

All (unordered, all must appear once):

```
<xs:element name="person">
  <xs:complexType>
    <xs:all>
      <xs:element name="firstName" type="xs:string"/>
      <xs:element name="lastName" type="xs:string"/>
      <xs:element name="age" type="xs:integer" minOccurs="0"/>
    </xs:all>
  </xs:complexType>
</xs:element>
```

XML Schemas

3.4.6 Cardinality with minOccurs/maxOccurs

Control how many times an element can appear:

```
<xs:element name="library">
   <xs:complexType>
      <xs:sequence>
         <!-- Exactly one (default) -->
         <xs:element name="name" type="xs:string"/>
         <!-- Zero or one (optional) -->
         <xs:element name="description" type="xs:string"
                  minOccurs="0"/>
         <!-- One or more -->
         <xs:element name="book" type="bookType"
                  minOccurs="1" maxOccurs="unbounded"/>
         <!-- Zero to five -->
         <xs:element name="featured" type="xs:string"
                  minOccurs="0" maxOccurs="5"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

XML Schemas

3.4.7 Attributes in Complex Types

```
<xs:element name="product">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="description" type="xs:string"/>
    </xs:sequence>
    <!-- Attribute definitions -->
    <xs:attribute name="id" type="xs:string" use="required"/>
    <xs:attribute name="category" type="xs:string" use="optional"/>
    <xs:attribute name="status" use="optional" default="active">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="active"/>
          <xs:enumeration value="discontinued"/>
          <xs:enumeration value="pending"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>
```

XML Schemas

3.4.8 Defining Reusable Types

Named Complex Types:

```
<xs:complexType name="addressType">
   <xs:sequence>
      <xs:element name="street" type="xs:string"/>
      <xs:element name="city" type="xs:string"/>
      <xs:element name="state" type="stateCodeType"/>
      <xs:element name="zip" type="xs:string"/>
   </xs:sequence>
   <xs:attribute name="country" type="xs:string" default="US"/>
</xs:complexType>
<xs:element name="shippingAddress" type="addressType"/>
<xs:element name="billingAddress" type="addressType"/>
```

XML Schemas

Type Derivation (Extension):

```
<xs:complexType name="internationalAddressType">
   <xs:complexContent>
      <xs:extension base="addressType">
         <xs:sequence>
            <xs:element name="country" type="xs:string"/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

XML Schemas

XML Schemas

XML Schemas