XML Schema

Outline

- XML Schema Overview
- XML Schema Components
- XML Schema Reusability & Conformance
- XML Schema Applications and IDE

XML Schema Overview

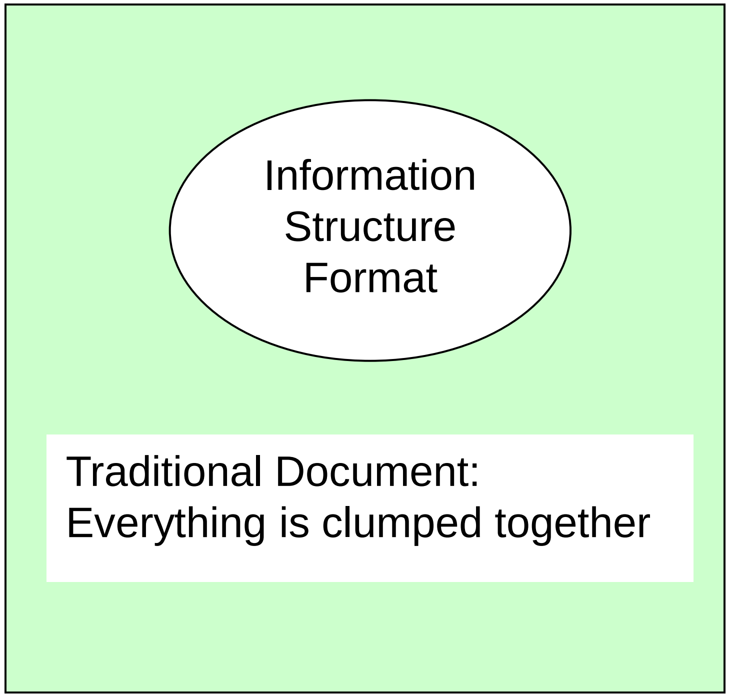
- What is XML Schema?
- Why Schema?
- A Simple XML Schema Example
- How to Convert DTD to Schema?

What is XML Schema?

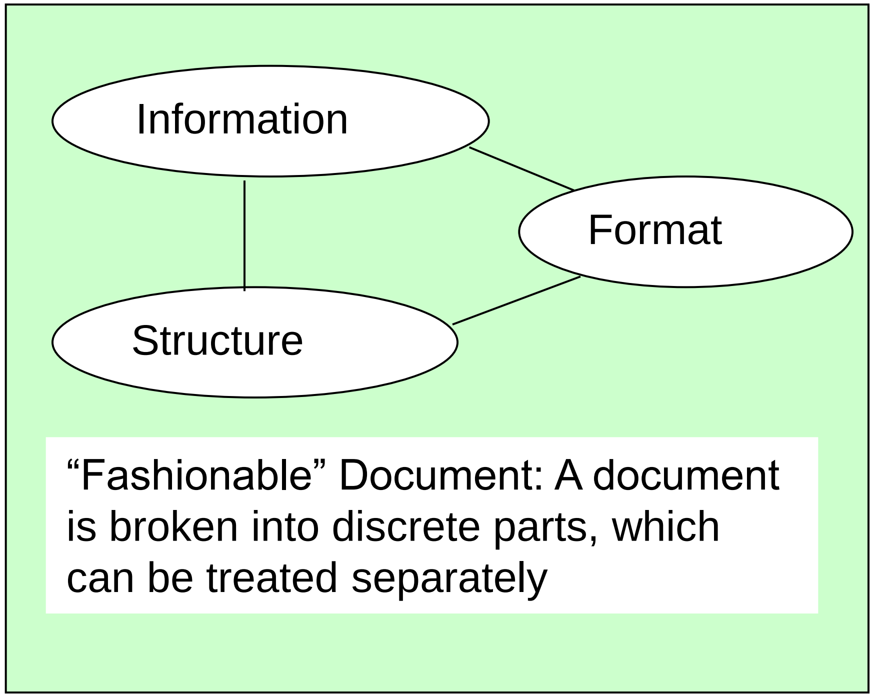
- The origin of schema
  - XML Schema documents are used to define and validate the content and structure of XML data.
  - XML Schema was originally proposed by Microsoft, but became an official W3C recommendation in May 2001

Why Schema? (1)

Separating Information from Structure and Format

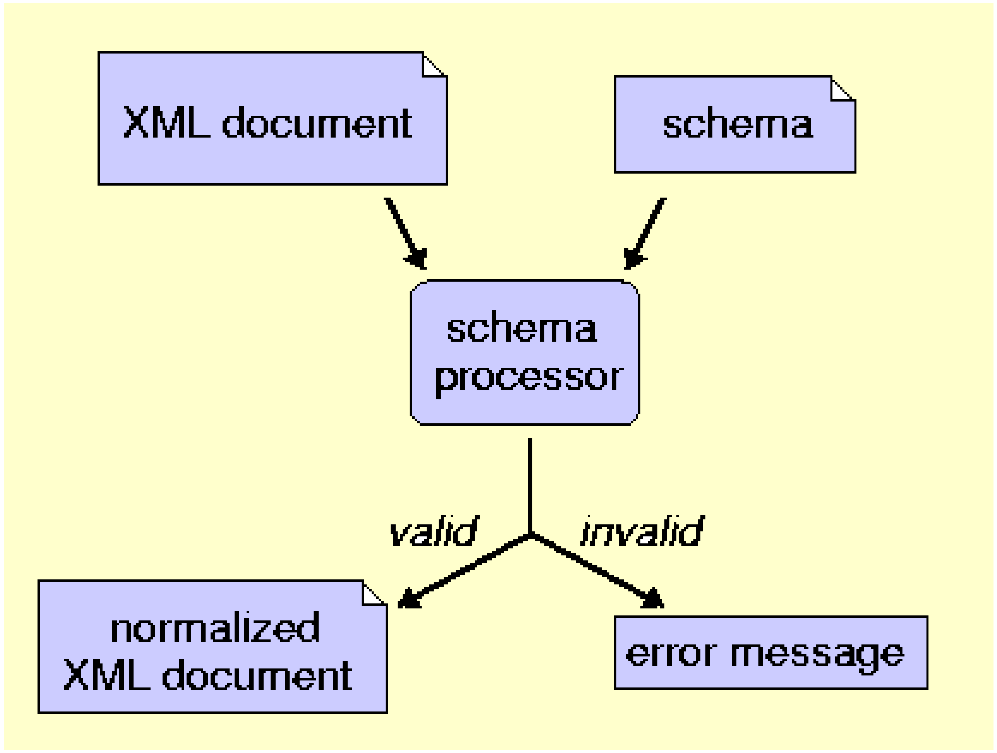


Traditional Document:
Everything is clumped together



“Fashionable” Document: A document is broken into discrete parts, which can be treated separately

Why Schema? (2)



Schema Workflow

DTD versus Schema

Limitations of DTD

- No constraints on character data
- Not using XML syntax
- No support for namespace
- Very limited for reusability and extensibility

Advantages of Schema

- Syntax in XML Style
- Supporting Namespace and import/include
- More data types
- Able to create complex data type by inheritance
- Inheritance by extension or restriction
- More …

Problems of XML Schema

- General Problem
  - Several-hundred-page spec in a very technical language
- Practical Limitations of expressibility
  - content and attribute declarations cannot depend on attributes or element context.
- Technical Problem
  - The notion of "type" adds an extra layer of confusing complexity
- …

An XML Instance Document Example

```
<book isbn="0836217462">
    <title> Being a Dog Is a Full-Time Job</title>
    <author>Charles M. Schulz</author>
    <qualification>    extroverted beagle  </qualification>
</book>
```

The Example’s Schema

```
<?xml version="1.0" encoding="utf-8"?>
    <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
      <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=“qualification“ type=“xs:string”/>
           </xs:sequence>
          </xs:complexType>
        </xs:element>
</xs:schema>
```

book.xsd

Outline

- XML Schema Overview
- XML Schema Components
- XML Schema Reusability & Conformance
- XML Schema Applications and IDE

Outline

- XML Schema Overview
- XML Schema Components
- XML Schema Reusability & Conformance
- XML Schema Applications and IDE

XML Schema Components

- Abstract Data Model
- Simple and Complex Type Definitions
- Declarations
- Relationship among Schema Components

XML Abstract Data Model

- The XML Abstract Data Model
  - composes of Schema Components.
  - is used to describe XML Schemas.
- Schema Component
  - is the generic term for the building blocks that compose the abstract data model of the schema.

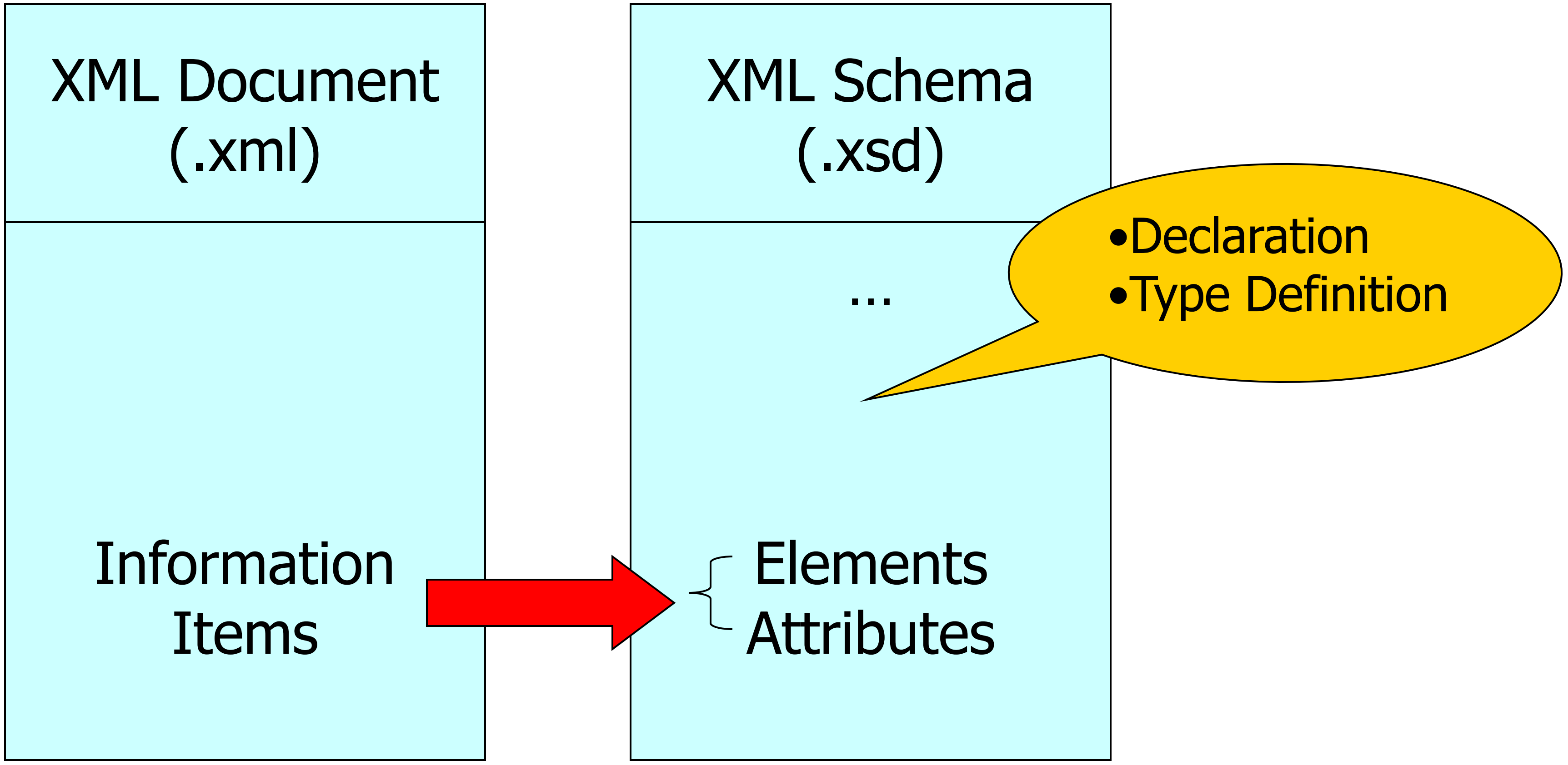
13 Kinds of Schema Components

- Simple type definitions
- Complex type definitions
- Attribute declarations
- Element declarations

- Attribute group definitions
- Identity-constraint definitions
- Model group definitions
- Notation declarations

- Annotations
- Model groups
- Particles
- Wildcards
- Attribute Uses

XML document & XML Schema



Declaration & Definition

- Declaration Components
  - are associated by (qualified) names to information items being validated.
  - It is like declaring objects in OOP.
- Definition Components
  - define internal schema components that can be used in other schema components.
  - Type definition is like defining classes in OOP.

Examples

```
<book isbn="0836217462">
 <title>
  Being a Dog Is a Full-Time Job
 </title>
……
</book>
```

Declaration

```
<xs:element name="book">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="title" type="xs:string"/>
              … …
    </xs:sequence>
    <xs:attribute name="isbn" type="xs:string"/>
  </xs:complexType>
</xs:element>
```

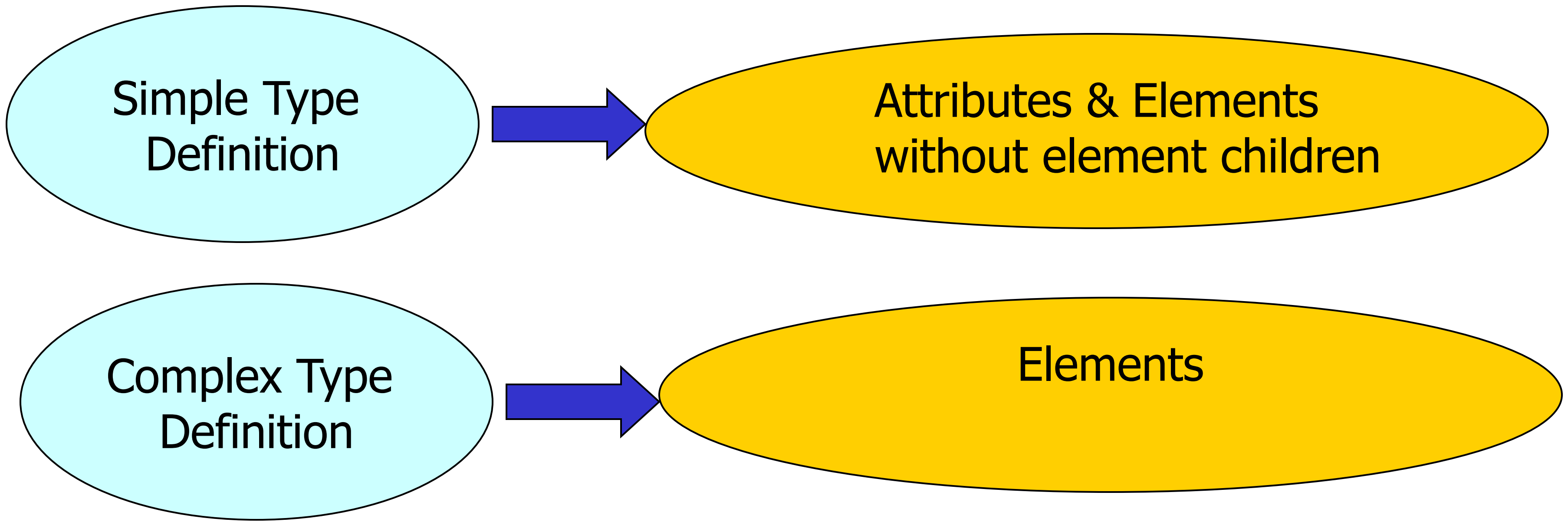
Type Definition

```
<xs:element name="book" type="bookType"/>
```

```
<xs:complexType name="bookType">
  <xs:sequence>
    <xs:element name="title" type="nameType"/>
      … …
  </xs:sequence>
  <xs:attribute name="isbn" type="isbnType" use="required"/>
</xs:complexType>
```

Type Definitions

- Why Type Definitions?
- Simple Type Definition VS. Complex Type Definition



Simple Type Definition

- Simple Type Definition can be:
  - a restriction of some other simple type;
  - a list or union of simple type definition;or
  - a built-in primitive datatypes.
- Example

```
<xs:simpleType name="farenheitWaterTemp">
 <xs:restriction base="xs:number">
  <xs:fractionDigits value="2"/>
  <xs:minExclusive value="0.00"/>
  <xs:maxExclusive value="100.00"/>
 </xs:restriction>
</xs:simpleType>
```

Complex Type Definition(1)

- Inheritance
  - Restriction: We restrict base types definitions.
  - Extension: We add something new.
- Composition
  - Group model

Complex Type Definition(2)

- Inheritance

  Each complex type definition is either
  - a restriction of a complex type definition
  - an extension of a simple or complex type definition
  - a restriction of the ur-type definition.
- Example

```
<xs:complexType name="personName">
 <xs:sequence>
  <xs:element name="title" minOccurs="0"/>
        … …
 </xs:sequence>
</xs:complexType>
```

```
<xs:complexType name="extendedName">
 <xs:complexContent>
  <xs:extension base="personName">
   <xs:sequence>
    <xs:element name="generation"
                     minOccurs="0"/>
   </xs:sequence>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
```

Complex Type Definition(3)

- Composition

  Model Group is composed of

  - Compositor (*sequence* | *choice*| *all*)
  - Particles, which can be
    - Element Declaration
    - Wildcard
    - Model Group

Complex Type Definition(4)

- Model Group Examples

```
<xs:all>
 <xs:element ref="cats"/>
 <xs:element ref="dogs"/>
</xs:all>
```

```
<xs:sequence>
 <xs:choice>
  <xs:element ref="left"/>
  <xs:element ref="right"/>
 </xs:choice>
 <xs:element ref="landmark"/>
</xs:sequence>
```

Complex Type Definition(5)

- Reusable Fragments of Type Definition
  - Model group definition
  - Attribute group definition

```
<xs:group name="mainBookElements">
  <xs:sequence>
    <xs:element name="title" type="nameType"/>
    <xs:element name="author" type="nameType"/>
  </xs:sequence>
</xs:group>
<xs:complexType name="bookType">
  <xs:sequence>
    <xs:group ref="mainBookElements"/>
      … …
  </xs:sequence>
  <xs:attributeGroup ref="bookAttributes"/>
</xs:complexType>
```

Declarations(1)

- Element Declaration
- Attribute Declaration
  - Attribute uses
- Notation Declaration
  - Notation declarations reconstruct XML 1.0 NOTATION declarations.

Declarations(2)

Examples

```
<xs:element name="PurchaseOrder" type="PurchaseOrderType"/>
```

```
<xs:element name="gift">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="birthday" type="xs:date"/>
   <xs:element ref="PurchaseOrder"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

```
<xs:attribute name="age" type="xs:positiveInteger" use="required"/>
```

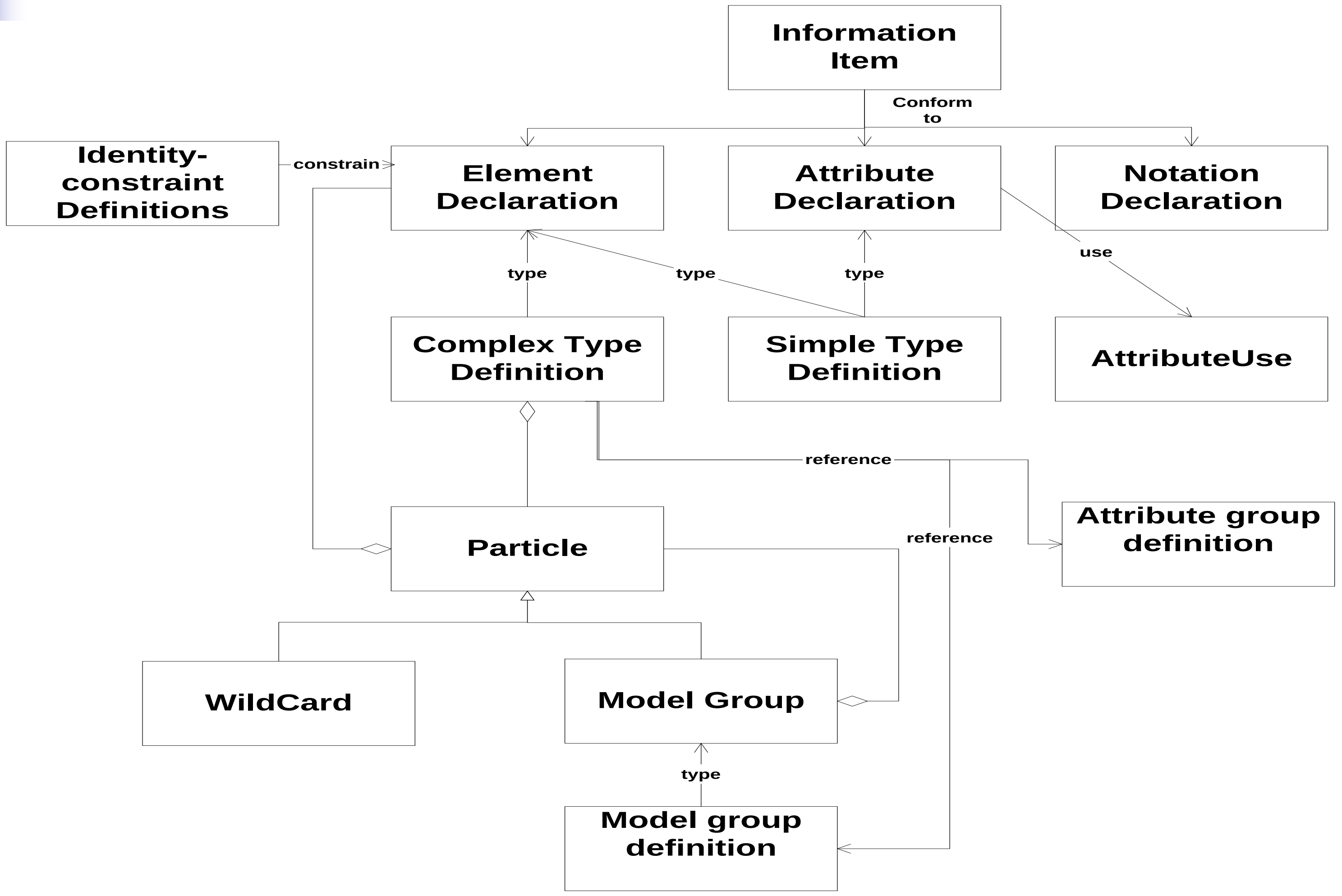
```
 <xs:notation name="jpeg" public="image/jpeg" system="viewer.exe">
```

Annotations

- Annotations provide for human- and machine-targeted documentations of schema components.
- They are meta-data.

```
<xs:simpleType fn:note="special">
  <xs:annotation>
   <xs:documentation>A type for experts only</xs:documentation>
   <xs:appinfo>
    <fn:specialHandling>checkForPrimes</fn:specialHandling>
   </xs:appinfo>
  </xs:annotation>
```

Relationships among Schema Components



Outline

- XML Schema Overview
- XML Schema Components
- XML Schema Reusability & Conformance
- XML Schema Applications and IDE

XML Schema
Reusability and Conformance

- XML Schema Reusability
- XML Schema Conformance

Building Reusable XML Schema

- Two mechanisms
  - Including and Redefining existing XML Schemas components in an XML Schema definition
  - Extending or Restricting existing data types in an XML Schema definition

Building Reusable XML Schema- (1)

- xs:include
  - Similar to a copy and paste of the included schema
  - The calling schema doesn't allow to override the definitions of the included schema.

```
<xs:schema …>
…
<xs:include schemaLocation="address.xsd"/>
<xs:include schemaLocation="items.xsd"/>
…
</xs:schema>
```

purchaseOrder.xsd

- xs:redefine
  - Similar to xs:include
  - except that it lets you redefine declarations from the included schema.

```
<xs:schema …>
…
<xs:redefine schemaLocation="chaper12.xsd">
   <xs:simpleType name="title">
      <xs:restriction base="xs:string">
         <xs:maxLength value="40"/>
      </xs:restriction>
   </xs:simpleType>
</xs:redefine>
…
</xs:schema>
```

chapter18.xsd

Building Reusable XML Schema-(2)

- Group head and Group members
  - Similar to Object-oriented design
  - Referencing a common element (called *head*) using substitutionGroup attribute
  - The *head* element is global and abstract. All the element within a substitution group need to have a type
    - the same type as *head* or
    - a derivation from it

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

  <xsd:element name="root">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref=“abstractInt"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name=“abstractInt" type=“decimal" abstract="true"/>
   <xsd:simpleType name=“decimal">
    <xsd:restriction base="xsd:integer">
      <xsd:minInclusive value="1"/>
      <xsd:maxInclusive value="9"/>
    </xsd:restriction>
   </xsd:simpleType>
  </xsd:element>
  <xsd:element name="odd" substitutionGroup=“abstractInt">
    <xsd:simpleType>
      <xsd:restriction base=“decimal">
        <xsd:enumeration value="1"/>
        <xsd:enumeration value="3"/>
        <xsd:enumeration value="5"/>
        <xsd:enumeration value="7"/>
        <xsd:enumeration value="9"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:element>

</xsd:schema>
```

Some invalid XML fragments:
1. <myAbstract>2</myAbstract>
2. <odd>2</odd>

Prohibit XML Schema Derivations

- The final attribute
  - Can be used in
    - xs:complexType
    - xs:simpleType
    - xs:element
  - Can take values of
    - restriction
    - extension
    - #all (any derivation)

```
<xs:complexType name="characterType" final="#all">
  <xs:sequence>
    <xs:element name="name" type="nameType"/>
    <xs:element name=“age" type=“ageType"/>
    <xs:element name="qualification" type="descType"/>
  </xs:sequence>
</xs:complexType>
```

- The fixed attribute
  - Can only be used in xs:simpleType
  - When it is set to true, it cannot be further modified

```
<xs:simpleType name=“fixedString">
  <xs:restriction base="xs:string">
    <xs:maxLength value="32" fixed="true"/>
  </xs:restriction>
</xs:simpleType>
```

What is XML Schema Conformance?

- Quoted from W3C XML Schema Structure:

  Any instance XML document may be processed against any schema to verify whether the rules specified in the schema are honored in the instance or not.

- However, the author of an instance XML document may claim to conform to a particular schema by using schemaLocation attribute

Conformance Checking for XML Instance Document

1. Check the root element has the right contents
2. Check each sub-element conforms to its description in a schema
3. And so on, until the entire document is verified.

Conformance Checking for an XML Element

1. Locate the declaration for this element in the XML schema
2. Examine the type of the element
3. Check the immediate attributes and contents of this element
4. Compare these values against the attributes and contents permitted by the element's type

Conformance Example: note.xml and note.xsd

```
<?xml version="1.0"? >
<note timestamp="2002-12-20">
 <to>Dove</to>
 <from>Jani</from>
 <heading>Reminder</heading>
 <body>Don't forget me this weekend!</body>
</note>
```

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" >
<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:attribute name="timestamp" type="xs:date" />
  </xs:complexType>
</xs:element>
</xs:schema>
```

XML Parsers

- Every XML application is based on a parser
- Two types of XML documents:
  - Well-formed:if it obeys the syntax of XML
  - Valid:if it conforms to a proper definition of legal structure and elements of an XML document
- Two types of XML Parsers:
  - Non-validating
  - Validating

Two Ways of Interfacing XML Documents with XML Applications

- Object-based: **DOM** (Document Object Model)
  - Specified by W3C
  - The parser loads the XML doc into computer memory and builds a tree of objects for all elements & attributes
- Event-based: **SAX** (Simple API for XML)
  - Originally a Java-only API.
  - Developed by XML-DEV mailing list community
  - No tree is built
  - The parser reads the file and triggers events as it finds elements/attribute/text in the XML doc

Available XML Schema-supported Parsers

- Apache® Xerces 2 Java/C++ free
  - Validating/Non-validating
  - DOM and SAX
- Microsoft® XML Parser 4.0 free
  - DOM and SAX
- TIBCO® XML Validate commercial
  - SAX-based implementation
  - Suitable in a streaming runtime environment
- SourceForge.net® JBind 1.0 free
  - A data binding framework linking Java and XML
  - Its Schema Compiler generates Java classes/interfaces for types contained in XML Schema.
  - The runtime environment is used to read/write XML documents for validation, accessing and manipulating XML data
- And many many more…

Outline

- XML Schema Overview
- XML Schema---A Case Study
- XML Schema Components
- XML Schema Reusability & Conformance
- XML Schema Applications and IDE

Remind Schema features

- Object-Oriented Features
  - Distinction between types and instances. Schema type definitions are independent of instance declarations.
  - Inheritance

- Relational information Features
  - Like tree structure; having parents and children
  - Strongly-typed: strong typing available in the specification.

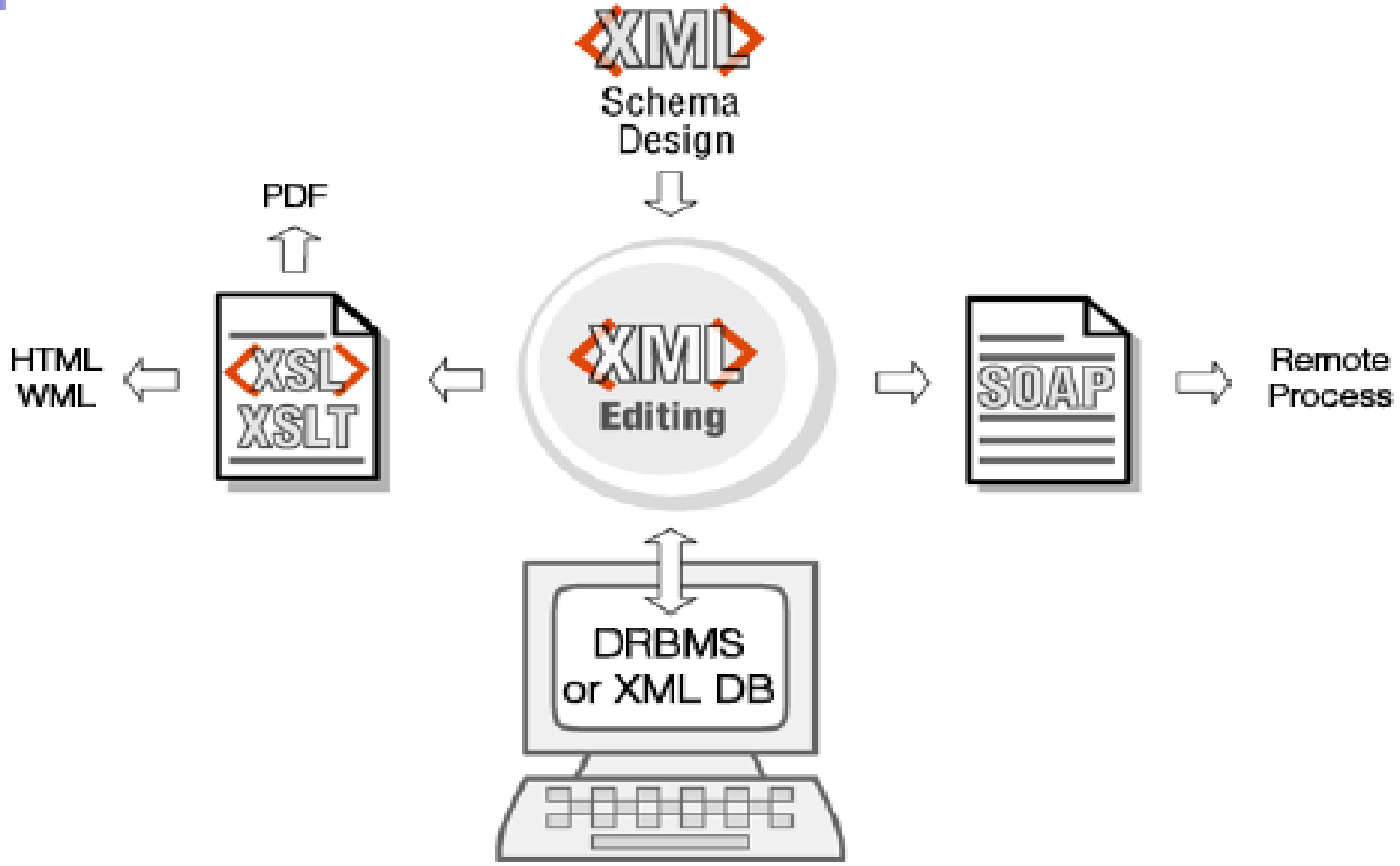
What is XML Software Development process? -(1)

1. Begin with developing content model using XML Schema or DTD

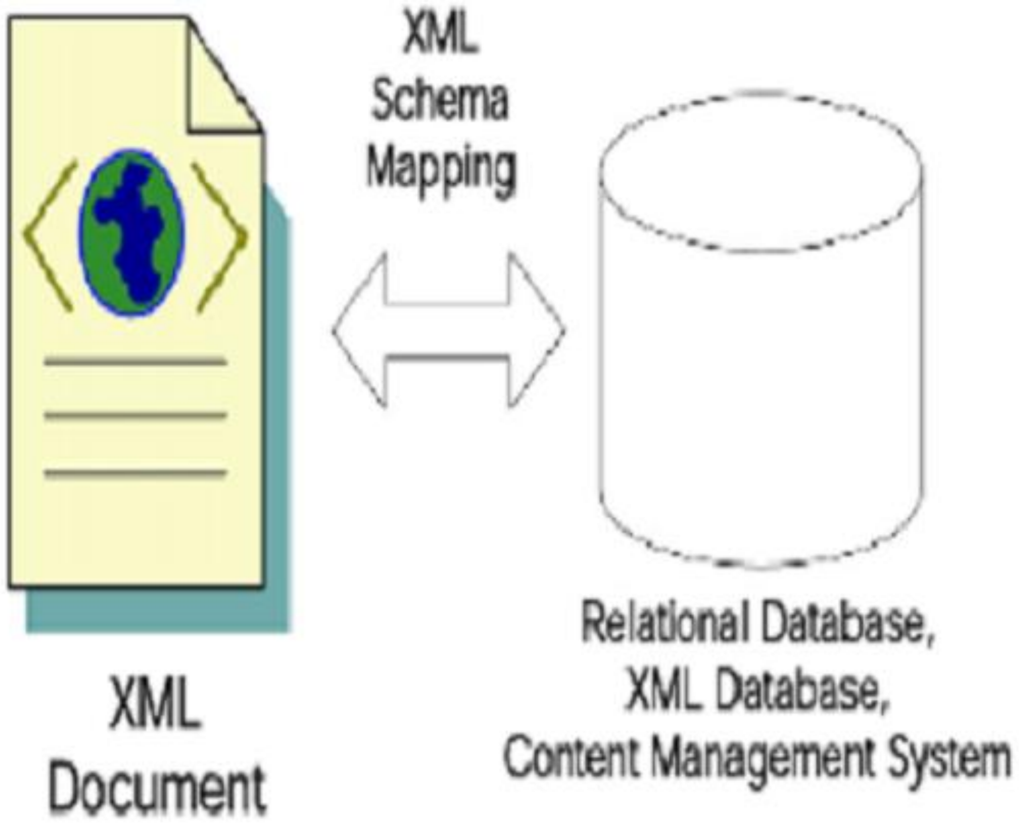
2. Edit and validate XML documents according to the content model

3. Finally, the XML document is ready to be used or processed by an XML enabled framework

What is XML Software Development process? -(2)

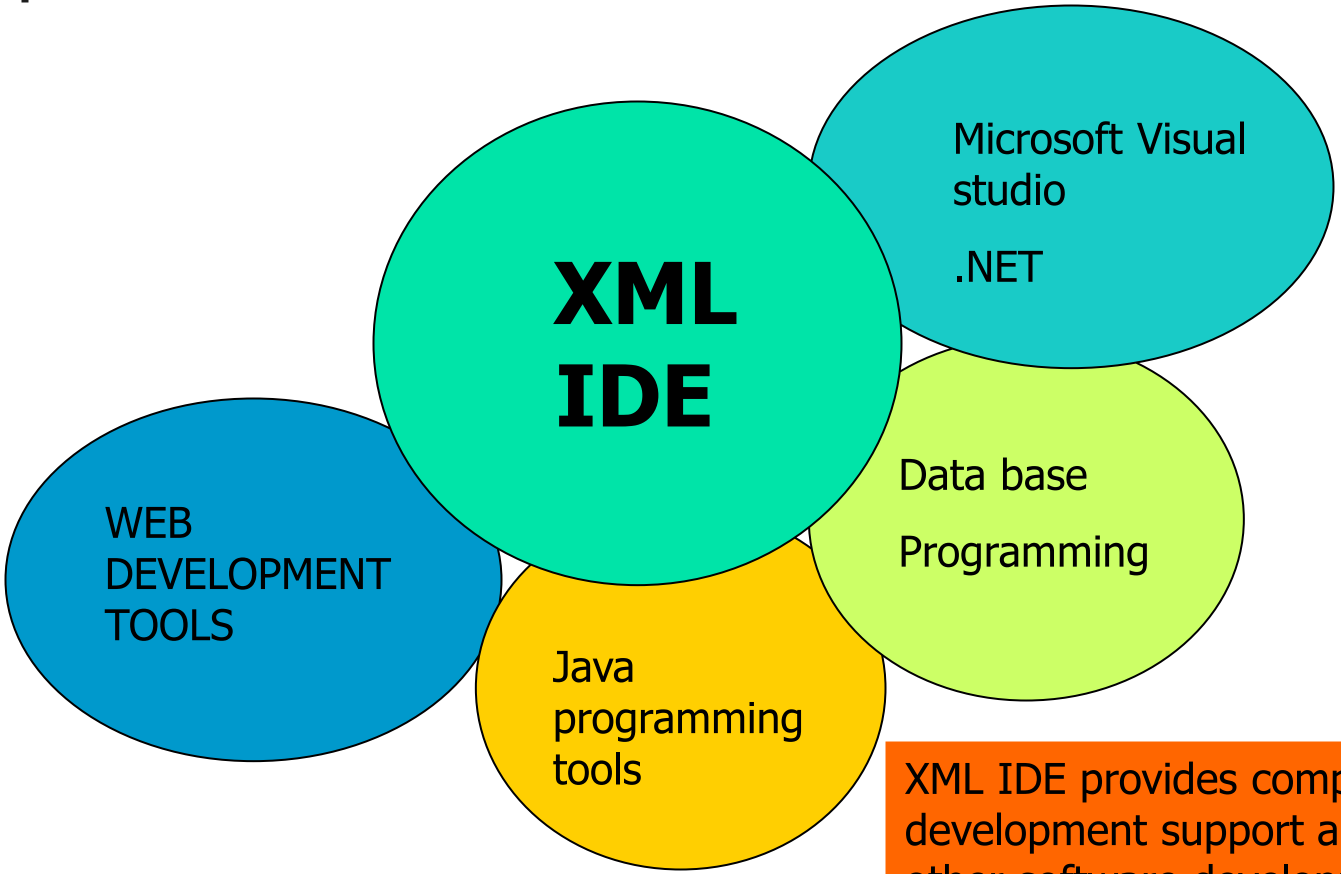


The xml software development process



Xml schema enable translations from XML documents to databases.

XML Integrated Development Environments (IDE) -1



XML IDE provides comprehensive XML development support and complements other software development tools

XML Integrated Development Environments (IDE)-2

- For example, XML Spy and Cape Clear Studio are both full XML IDEs.

References

[1] http://www.cs.concordia.ca/~faculty/haarslev/teaching/semweb/XMLschema.ppt

[2] *XML Bible, 2nd Edition* http://www.ibiblio.org/xml/books/bible2/index.html

Summary

- XML Schema Overview
- XML Schema---A Case Study
- XML Schema Components
- XML Schema Reusability & Conformance
- XML Schema Applications and IDE

Questions?