

# 0301402 INTRODUCTION TO XML

UNIT	MODULES	WEIGHTAGE
1	Introduction to XML	20 %
2	Document Type Definition (DTD)	20 %
3	XML Namespace	20 %
4	XML Schema	20 %
5	Extensible StyleSheet Language (XSL)	20 %

# UNIT - 4 XML Schema

- Introduction to Schema
- Features
- DTD versus XML Schema
- XML Schema Type System
  - Simple Types
  - Complex Types
- Grouping of Data
- Deriving Types
- Attributes

# Introduction to Schema

- A schema is **an alternative to DTD**.
- DTD are easier to write and provide supports for some feature, **but schemas are far richer in terms of their capabilities and extensibility**.
- The main difference between a DTD and a schema is that the **syntax of a DTD is different from that an XML. But syntax of a schema as that of an XML**.

# Introduction to Schema

- **DEMO**
  - message.xml
  - message.xsd
- XML schema is defined in a separate file.
- This file has the extension **.xsd**

# Introduction to Schema

- Message.xml

These explanations are depicted in Figure 4.3.

```
<?xml version = "1.0" ?>
```

This is normal XML declaration. There is nothing unusual or unique about this.

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

MESSAGE: This is the root element.

xmlns is the XML schema reference for our schema.

xsi:noNamespaceSchemaLocation provides a pointer to our schema. In this case, it is message.xsd.

Hello World!

```
</MESSAGE>
```

This is also nothing unusual. We simply specify the contents of our root element, and then signify the end of the root element (and hence that of the XML document).

**Figure 4.3** Understanding our XML document



# Introduction to Schema

- Message.xsd

These explanations are depicted in Figure 4.4.

```
<?xml version = "1.0" ?>
```

This is normal XML declaration. There is nothing unusual or unique about this.

```
<xsd:schema xmlns:xsd = "http://www.w3org/2001/XMLSchema">
```

xsd:schema indicates that this is a schema definition. xsd is the namespace prefix. It is associated with an actual namespace URI <http://www.w3org/2001/XMLSchema>.

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

This declares that our XML document will have the root element named MESSAGE of type string.

```
</xsd:schema>
```

This signifies the end of our schema file.

**Figure 4.4** Understanding our XML schema

## Introduction to Schema – class work

- Write an XML document that contains a single element to specify the name of the student. Provide a corresponding XML Schema
- DEMO
  - Student.xml
  - Student.xsd

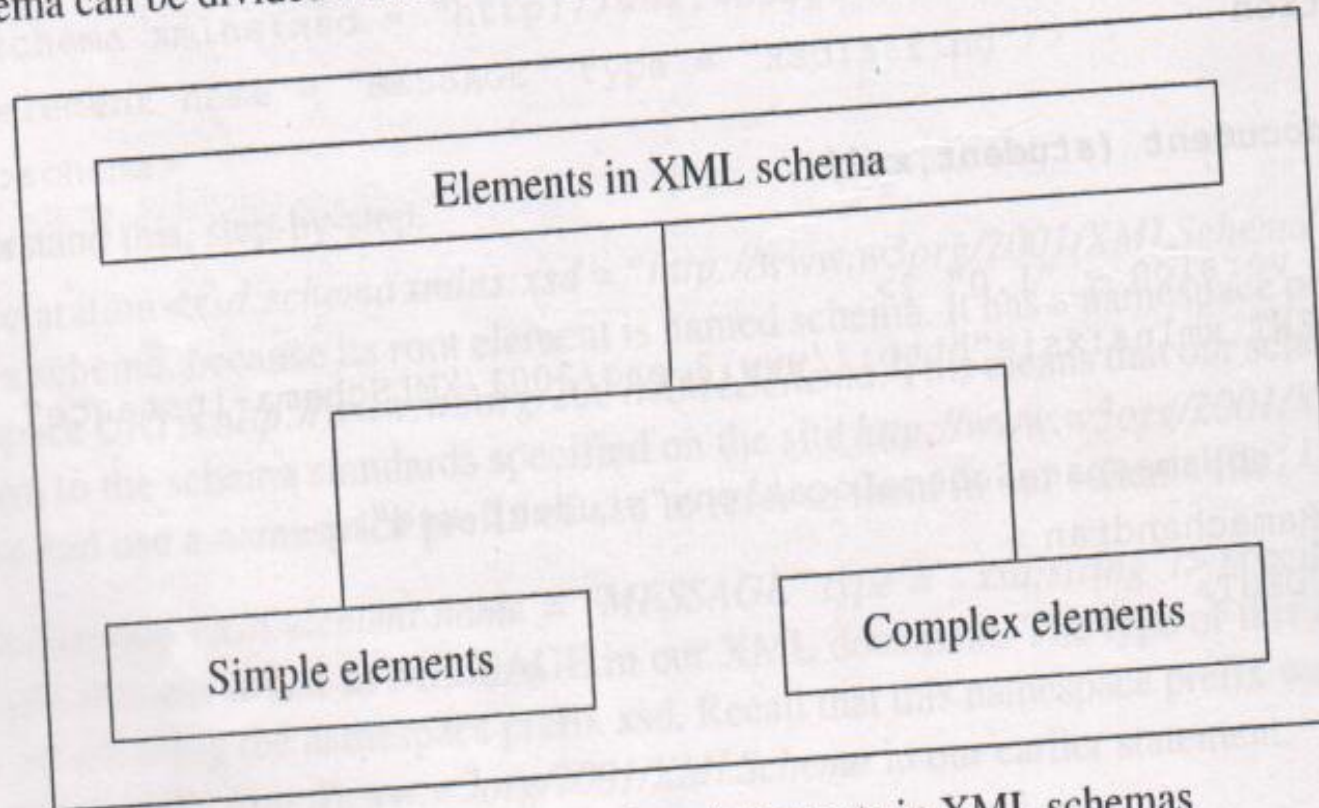
# XML Schema Type System

- Element in schema can be divided into two categories :
  - Simple Elements
  - Complex Elements
- **Simple Element** can contain only text. They cannot have sub-element or attributes.
- **Complex Element** can contain sub elements, attributes etc.



# XML Schema Type System

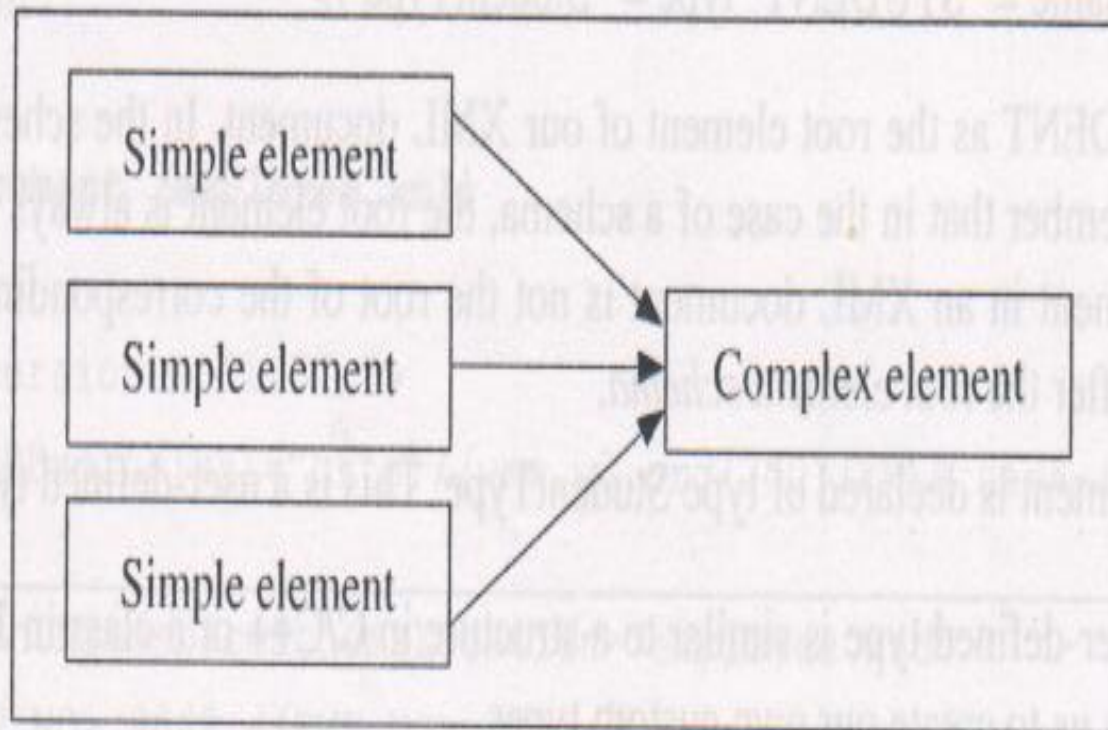
Elements in schema can be divided into two categories



**Figure 4.5** Classification of elements in XML schemas

# Complex Types

- Complex element is made up of simple elements



**Figure 4.6** Complex element is made up of simple elements

# Complex Types

- **Complex Type Demo**
  - Student1.xml
  - Student1.xsd

## Complex Types – class work

- Write an XML document and a corresponding XML schema for maintaining the employee number, name, designation and salary.
- DEMO
  - employee.xml
  - employee.xsd

## Specifying the Frequency

- If we want to repeat the information in XML then we must use either :
  - **minOccurs** attribute
  - **maxOccurs** attribute
- **minOccurs** attribute specifies the minimum number of occurrences that the element have. The default vale is 1.
- **maxOccurs** attribute specifies the maximum number of occurrences that the element have. The default vale is 1.

# Specifying the Frequency

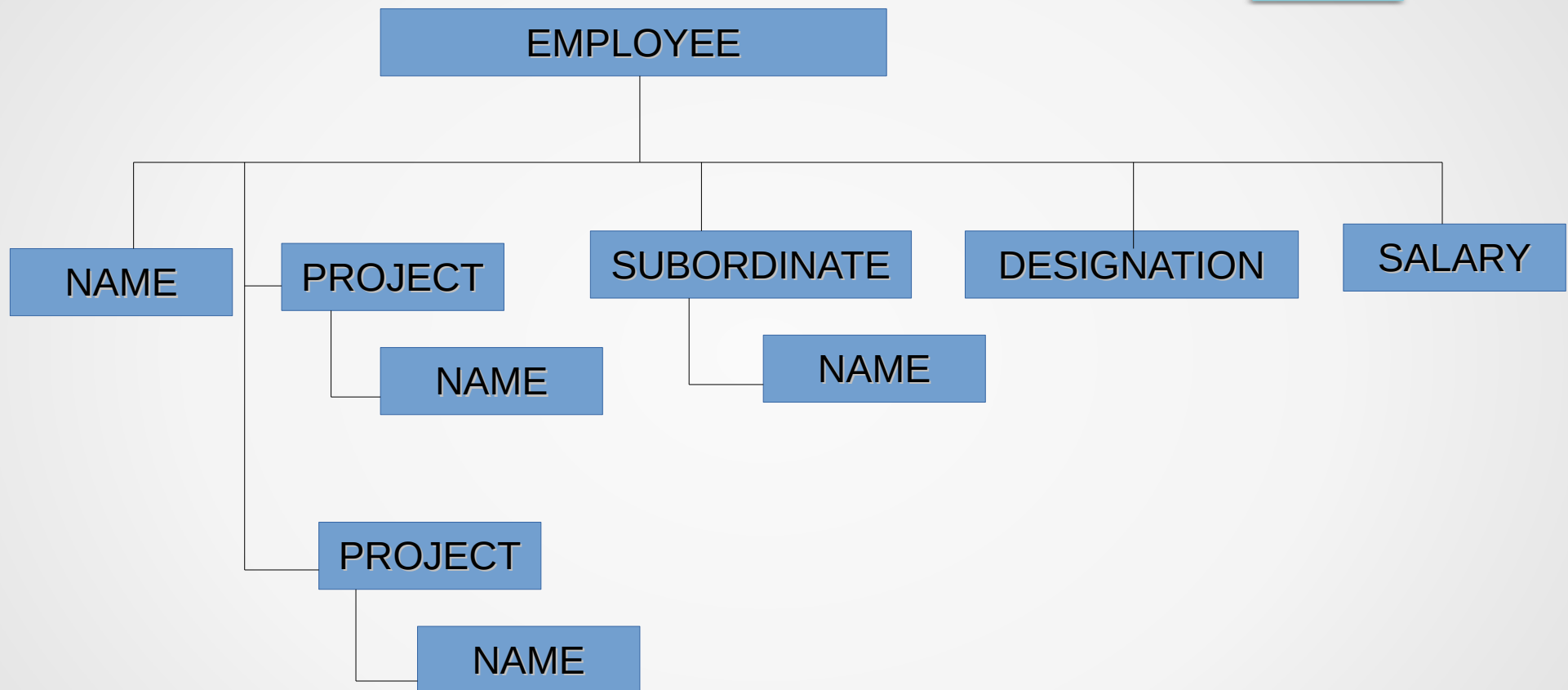
- DEMO
  - book.xml
  - bool.xsd



# Specifying the Frequency

Requirement	Set minOccurs to	Set maxOccurs to
An element should occur exactly once	1	1
An element should occur at least once and possibly many more time	1	unbounded
An elemet is optional or may occur for any number of time	0	unbounded
An element may not occur at all, or may occur only once	0	1

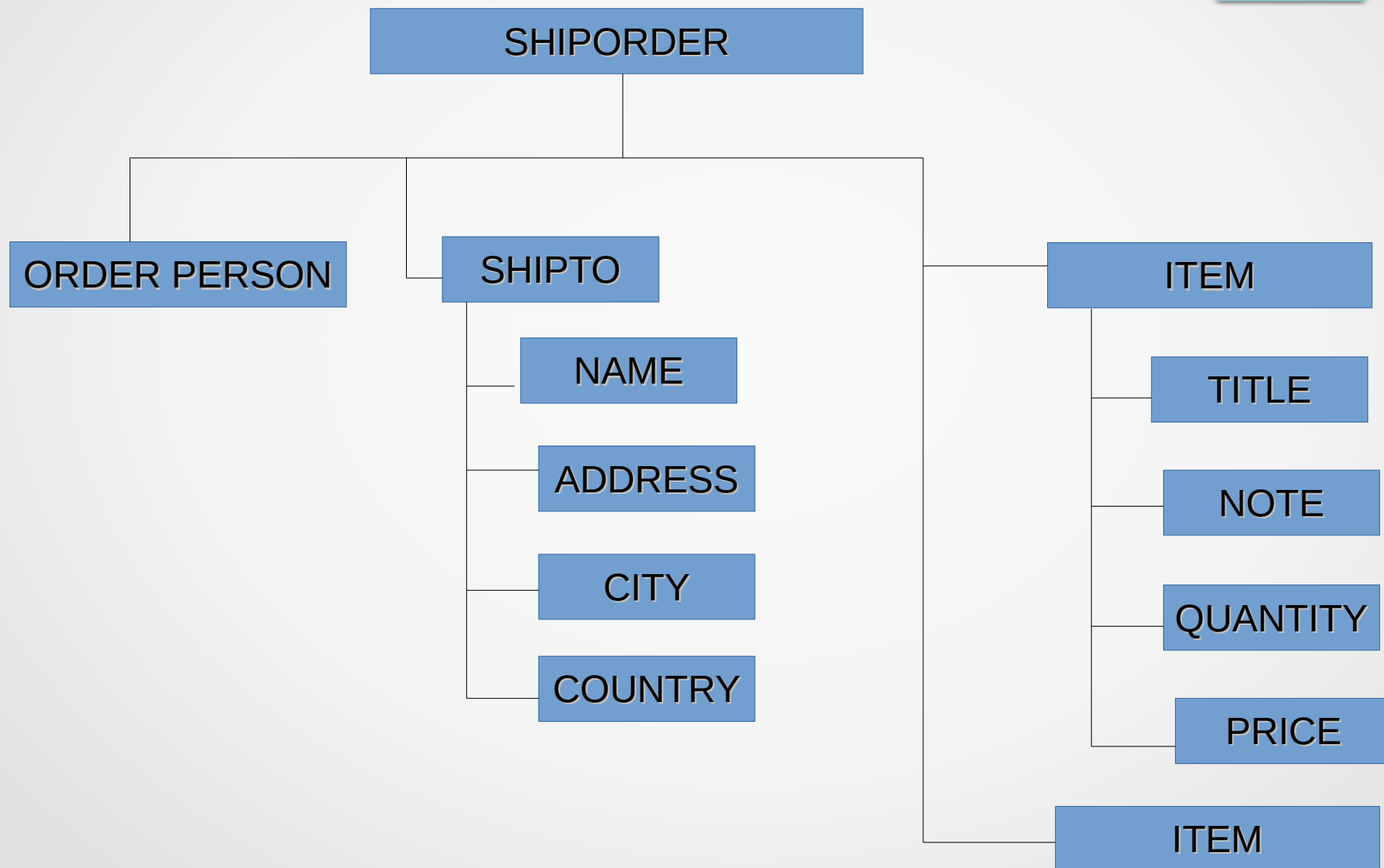
# Specifying Element content



# Specifying Element content

- DEMO
  - Emp.xml
  - Emp.xsd

# Specifying Element content



# Specifying Element content

- DEMO
  - shiporder.xml
  - shiporder.xsd

# Mixed Content

- Sometime we want to allow text between elements.
- For example we want to capture information about employee names. The elements are
  - Title
  - First Name
  - Last Name
  - Middle Name
  - Here we can ignore “Title” and “Middle Name”
- For this we can use “Mixed Content”



# Mixed Content

- Demo
  - Emp\_mix.xml
  - Emp\_mix.xsd

# Grouping Of Data

- Some times we just want to make sure that an element **exists inside an XML document** – where, is not so important.
- For this we can use the concept of “Grouping Element”
- Schema syntax provide supports for **three grouping constructs** :
  - **Xsd:all**
  - **Xsd:choice**
  - **Xsd:sequence**

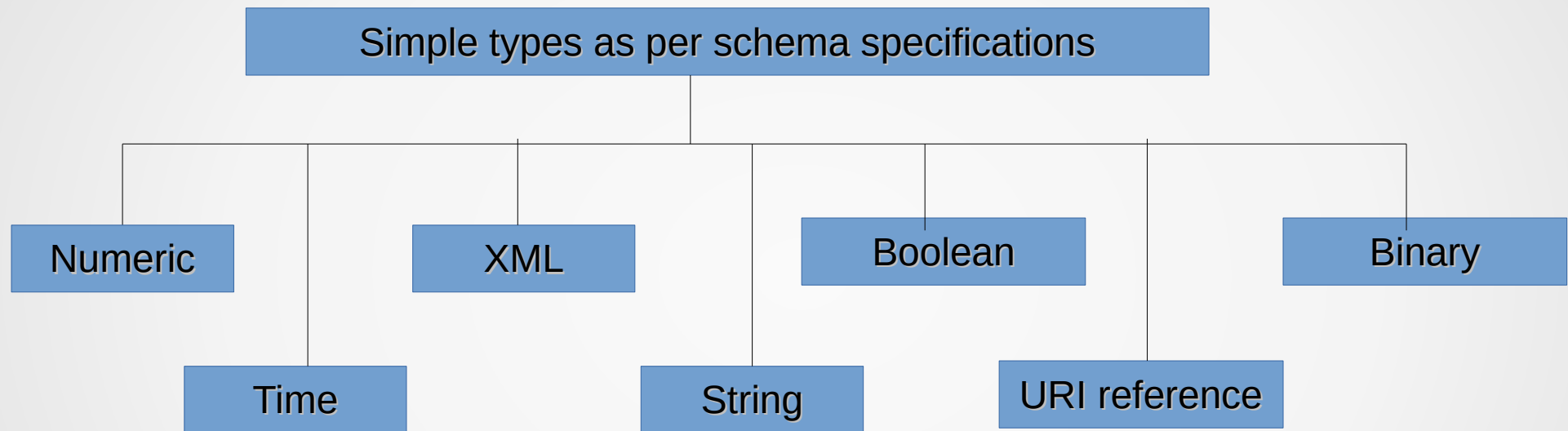
## Grouping Of Data

- **Xsd:all** - specifies that **all the elements** in a group **must occur** at the most once, but their **ordiering** is not significant.
- **Xsd:choice** - allows us to specify that **only one element** from the group can appear.
  - Alternatively, we can also specify that **out of n elements in a group, m should appear in any order.**
- **Xsd:sequence** – it mandates that every **element in a group must appear exactly once** and also in the same order in which the elements are listed.

# Grouping Of Data

- Demo
  - **xsd:all**
    - card.xml
    - card.xsd
  - **xsd:choice**
    - Result.xml
    - Result.xsd

# Simple Types



- XML schemas **offer 44 built-in simple types.**
- The XML schema simple type is **classify into seven categories.**

## Simple Type – Numeric Data Types

Data Type Name	Meaning	Example
xsd:float	32 – bit float type	0, 12345.2356
xsd:double	64 – bit double type	0,45.89E-2
xsd:decimal	Arbitrary precision (Big Decimal)	87200.29, -3.124578926
xsd:integer	Arbitrary large or small number	-789253598138965, 24535989326475
xsd:nonPositiveInteger	Integer less than or equal to 0	0, -1, -2



# Simple Type – Numeric Data Types

Data Type Name	Meaning	Example
xsd:negativeInteger	Integer less than 0	-1,-2,-3
xsd:nonNegativeInteger	Integer greater than or equal to 0	0,1, 2 ,3
xsd:positiveInteger	Integer greater than 0	1,2,3
xsd:long	8 byte 2's complement integer	2356985668522
xsd:int	4 byte 2's complement integer	-615251, 0, 125369

# Simple Type – Numeric Data Types

Data Type Name	Meaning	Example
xsd:short	2 – byte , 2's complement integer	-32767 to +32767
xsd:byte	1 – byte 2's complement integer	-128 to +127
xsd:unsignedLong	8 byte unsigned long	
xsd:unsignedInt	4 byte unsigned integer	
xsd:unsignedShort	2 byte unsigned integer	
xsd:unsignedByte	1 byte unsigned integer	

# Simple Type – Time Data Types

Data Type Name	Meaning	Example
xsd:dateTime	Date and time in the format YYYY-MM-DDTHH:MM:SS	2006-02-25T06:05:33
xsd:date	YYYY-MM-DD	
xsd:time	HH:MM:SS	
xsd:gDay	A day in a month	--01, --23
xsd:gMonth	A month in a year	--02--, --05--
xsd:gYear	A Year	2006

## Simple Type – Time Data Types

Data Type Name	Meaning	Example
Xsd:YearMonth	A pecific month in a specific year	2006-02, 2017-05
xsd:gMonthDay	A date without year	--01-20
xsd:duration	Length of time in format	P2006Y01M20DT06H11M03S

# Simple Type – XML Data Types

Data Type Name	Meaning	Example
xsd:ID	A unique value for an element or an attribute	T1, M90, G101
xsd:IDREF	Value of another ID type defined elsewhere in the document	
xsd:ENTITY	An XML name, declared as an unpaired entity in DTD	
xsd:NOTATION	Usually indicates a file format	
xsd:IDREFS	Reference to a list of ID names	
xsd:ENTITIES	List of entity names	

# Simple Type – XML Data Types

Data Type Name	Meaning	Example
xsd:NMTOKEN	NMTOKEN type	
xsd:NMTOKENS	A list of NMTOKEN types	
xsd:language	Language name from a list of valid value	
xsd:Name	XML name with or without colons	Student, emp
xsd:QName	Prefixed name	
xsd:NCName	Local name without colons	



## Simple Type – String Data Types

Data Type Name	Meaning	Example
xsd:string	A unicode character based string of any length.	
xsd:normalized String	A string in which all the carriage returns, linefeeds, and tabs are replaced with a single blank (space) character.	
xsd:token	Same as above, but in addition all leading and trailing spaces are trimmed and consecutive spaces are converted into a single space.	

## Simple Type – Binary Data Types

- XML support binary data types.
- But the problem with binary data is that it can have byte patterns that are illegal.
- **This is because some characters such as null have a different meaning, and they cannot be a part of the XML content.**
- We need to encode such illegal characters into a legal form
  - By hexadecimal conversion
  - By base-64 encoding

## Simple Type – Other Data Types

- **Boolean Data Type**

- xsd:boolean

- it allows one of the four possible value : zero, one, true and false.

- **URI Data Type**

- xsd:URI

- It allows us to specify a URI

- For i.e – <http://www.test.com/name.html>

# Deriving Types

- There are three technique for deriving types

Deriving simple types in XML schemas

```
graph TD; A[Deriving simple types in XML schemas] --> B[Restriction]; A --> C[Union]; A --> D[List];
```

Restriction

Union

List

# Deriving Types - Restriction

- **Restriction**

- Restriction allows us to select a subset of values allowed by the base type.
- We can use an element of type `xsd:restriction` as a child element of an `xsd:simple` type element.

# Deriving Types - Restriction - Facets

- **Restriction - Facets**

- A facet allows us to specify more restrictions than what a basic type allows.

Facet	Description
xsd:minInclusive	The minimum value that all the instances of this type must be greater than or equal to
xsd:maxInclusive	The maximum value that all the instances of this type must be less than or equal to
xsd:minExclusive	The minimum value that all the instances of this type must be greater than
xsd:maxExclusive	The maximum value that all the instances of this type must be less than
xsd:enumeration	A list of allowed values
xsd:whiteSpace	How white space is treated in this element

# Deriving Types - Restriction - Facets

- **Restriction - Facets**

- A facet allows us to specify more restrictions than what a basic type allows.

Facet	Description
xsd:pattern	A pattern with which the contents of the element are compared
xsd:length	The length of a string, items in a lis, or bytes in binary data
xsd:minLength	The minimum length
xsd:maxLength	The maximum length
xsd:totalDigits	The maximum number of digits allowed in the element
xsd:fractionDigits	The maximum number of digits allowed in the fractional part of the element.

## Deriving Types - Restriction – String

- **Restriction**
  - Xsd:length, xsd:minLength and xsd:maxLength these three facets are used for string.
- **Demos**
  - **book\_FACET.xml**
  - **book\_FACET.xsd**
  
  - **Emp\_FACET.xml**
  - **Emp\_FACET.xsd**



# Deriving Types - Restriction – White space

- **Restriction - String**

- The white space facet allows us to specify how we want to deal with white spaces.
- The `xsd:whiteSpace` facet allows three possible values,
  - **preserve**: This is the default. It means that the white space in XML doc. Kept as it is.
  - **replace** : replace every Tab, line feed, carriage return character in XML doc. With a single space character.
  - **collapse** : This facet value is a superset of the replace value. After performing the job of replace, this facet value further condenses multiple consecutive spaces into a single space.

# Deriving Types - Restriction – White space

- **Demos**
  - **Poem\_facet.xml**
  - **poem\_facet.xsd**

# Deriving Types - Restriction – Enumeration

- **Restriction - Enumeration**

- The enumeration facet in XML schemas allows us to specify a list of possible values for an element.

```
<xsd:simpleType name="BookCategory">  
  <xsd:restriction base="xsd:string">  
    <xsd:enumeration value="computer archi"/>  
    <xsd:enumeration value="network"/>  
  </xsd:restriction>  
</xsd:simpleType>
```

# Deriving Types - Unions

- Unions allow us to combine simple types to create new simple type.
- **For i.e**
  - Studnet can identify by his/ her Roll No. And /or Name
  - So we can create Union type which have either roll no or the student name.
- **Demos**
  - **union\_student.xml**
  - **union\_student.xsd**

## Deriving Types - Lists

- A list type allows the creation of a list of a particular simple type.
- **For i.e**
  - `<emp_list> 2562 2365 3698 4563 7821 </emp_list>`
- **Demos**
  - `list_emp.xml`
  - `list_emp.xsd`

# Attributes

- `<attribute name="des" type="xsd:type" use=" " />`
- An attribute occurs only once.
- We can specify whether
  - **Required** : must have
  - **Optional** : may have an attribute
  - **Prohibited** : cannot have an attribute
- i.e

`<attribute name="designation" type="xsd:string" use="required" default="Manager"/>`

# Grouping Attributes

- If an element have several attributes, then we can group them and provide a reference of this group to the concerned element.

# Grouping Attributes

```
<xsd:element name="EMPLOYEE">  
  <xsd:complexType>  
    <xsd:attributeGroup ref="empDetails"/>  
  </xsd:complexType>  
</xsd:element>
```

```
<xsd:attributeGroup name="empDetails">  
  <xsd:attribute name="empID" use="required" type="xsd:ID"/>  
  <xsd:attribute name="name" use="required" type="xsd:string"/>  
  <xsd:attribute name="designation" use="optional" type="xsd:string"/>  
</xsd:attributeGroup>
```



# Features of Schema

- XML schema uses XML instance syntax
- XML schema allows a rich variety of data types to be used to constrain both element and attribute content.
- XML schema allows us to specify which namespace declarations and definitions belongin.
- In XML schema, type definitions and element and attribute declarations are separated from each other.
- XML schema allows us to specify constraints on the uniqueness of values of a particular type, as well as relationships between those unique values and value of other type.