

Schéma XML

- DTDs et Schémas XML
- Exemple
- Élément schéma
- Types simples
- Types complexes
- Inclusion de schémas

Références

<http://www.w3.org/TR/xmlschema-1/> (Structures)

<http://www.w3.org/TR/xmlschema-2/> (Datatypes)

Objectifs des DTDS et schéma

- Définir la structure d'un document indépendamment de sa représentation physique et visuelle
- Définir le domaine des éléments et des attributs

Limites des DTDs

1. Syntaxes non xml

=> deux langages distincts

2. typage faible

=> PCDATA, pas de type string, integer, decimal, date

3. contraintes de cardinalités, de domaine non exprimables

=> uniquement références croisées

4. Non extensible

Expression des contraintes

1. Le lieu (location) doit comprendre une latitude suivie d'une longitude
2. La latitude doit être un décimal dont la valeur est comprise entre -90 et +90
3. La longitude doit être un décimal dont la valeur est comprise entre -180 et +180
4. Pour les deux éléments la précision est de 6 chiffres après la virgule

```
<location>  
<latitude>32.904237</latitude>  
<longitude>73.620290</longitude>  
</location>
```

Validation d'un document xml

```
<location>  
<latitude>32.904237</latitude>  
<longitude>73.620290</longitude>  
</location>
```

Document xml

Validation
XML Schema

Document valide

- contrôle que la latitude est entre -90 and +90
- contrôle que la longitude est entre -180 and +180
- contrôle que la précision est de 6 chiffres après la virgule

XML Schema

Caractéristiques d'un schéma XML

- Types de données
 - +44
 - Création de ses propres types de données

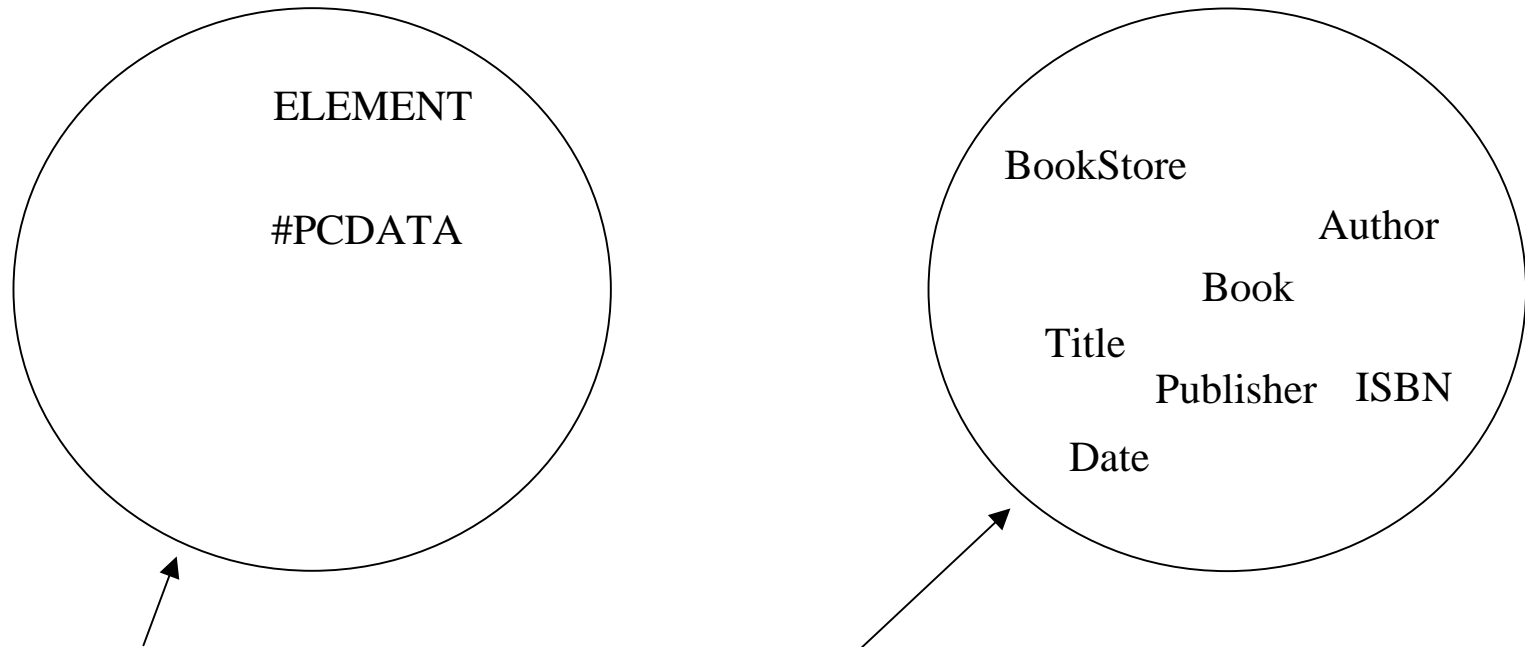
Type basé sur le type *string* et dont les éléments doivent être formés de la chaîne dd-dd-dd-dd-dd où d représente un chiffre .
 - Même syntaxe que les documents
 - Moins de syntaxe à se rappeler
 - Orienté objet
 - Possibilité d'étendre ou de restreindre un type

(Définition d'un nouveau type sur la base d'un ancien)
 - Expression d'unicité d'un élément
 - Définition d'éléments multiples avec le même nom mais des contenus différents
 - Définition d'un élément vide
 - Définition d'éléments substituables
- (élément *livre* peut être remplacé par l'élément *publication*)

BookStore.dtd

```
<!ELEMENT BookStore (Book+)>  
<!ELEMENT Book (Title, Author, Date, ISBN, Publisher)>  
<!ELEMENT Title (#PCDATA)>  
<!ELEMENT Author (#PCDATA)>  
<!ELEMENT Date (#PCDATA)>  
<!ELEMENT ISBN (#PCDATA)>  
<!ELEMENT Publisher (#PCDATA)>
```

Vocabulaire de la DTD

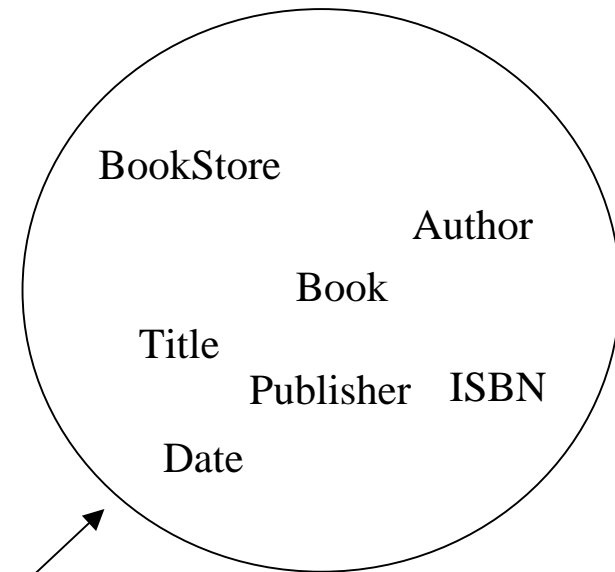
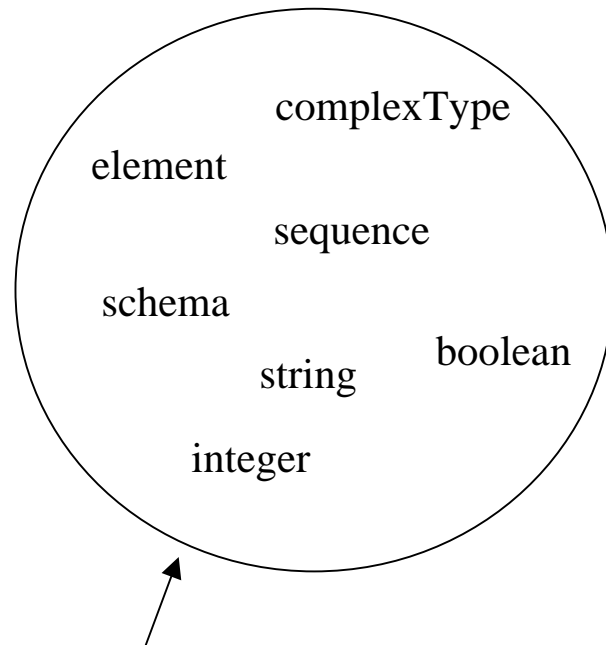


C'est le vocabulaire fournit par la DTD
pour définir les éléments associé
aux documents d'une librairie

Vocabulaire d'un schéma XML

<http://www.w3.org/2001/XMLSchema>

<http://www.books.org> (*targetNamespace*)



Vocabulaire fournit par XML schema
pour définir notre nouveau
vocabulaire

Le vocabulaire *XML Schema* est associé à un nom (espace de noms)
comme le nouveau Vocabulaire qui doit être associé à un nom.

Fichier BookStore.xsd

xsd = Xml-Schema Definition

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

```
  <xsd:element name="BookStore">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Book" minOccurs="1" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
```

<!ELEMENT BookStore (Book+)>

```
  <xsd:element name="Book">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Title" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Author" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Date" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="ISBN" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Publisher" minOccurs="1" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
```

<!ELEMENT Book (Title, Author, Date, ISBN, Publisher)>

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

<!ELEMENT Title (#PCDATA)>

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

<!ELEMENT Author (#PCDATA)>

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

<!ELEMENT Date (#PCDATA)>

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

<!ELEMENT ISBN (#PCDATA)>

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

<!ELEMENT Publisher (#PCDATA)>

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

Élément schéma

Élément racine de tout schéma XML

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

Éléments et types de données qui sont utilisés pour construire les schémas

- schema
- element
- complexType
- sequence
- string

viennent de l'espace de noms

<http://.../XMLSchema>

Espace de noms cible

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

xmlns="http://www.books.org"
elementFormDefault="qualified">

...

</xsd:schema>
```

Indique que les éléments
définis par ce schéma

- BookStore
- Book
- Title
- Author
- Date
- ISBN
- Publisher

sont dans l'espace de
noms <http://.../books.org>

Espace de noms par défaut

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

L'espace par défaut est
<http://www.books.org>
qui est l'espace de
noms cible!

Directive pour toute instance
document conforme à ce
schéma:

Tout élément utilisé par un
document instance qui est
déclaré dans ce schéma doit
être qualifié par l'espace de
noms

Espace de noms par défaut (2)

```
<xsd:element name="BookStore">
```

```
<xsd:complexType>
```

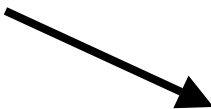
```
<xsd:sequence>
```

```
<xsd:element ref="Book" minOccurs="1" maxOccurs="unbounded"/>
```

```
</xsd:sequence>
```

```
</xsd:complexType>
```

```
</xsd:element>
```



Référence de la déclaration de l'élément Book. Mais dans quel espace de noms ?

En l'absence d'espace de noms, c'est l'espace de noms par défaut qui est considéré

Ainsi c'est une référence à la déclaration de l'élément Book de ce schéma.

Référence du schéma associé à une instance document XML : bookstore.xml

```
<?xml version="1.0"?>
<BookStore xmlns="http://www.books.org" ①
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.books.org BookStore.xsd">
  <Book>
    <Title>My Life and Times</Title>
    <Author>Paul McCartney</Author>
    <Date>July, 1998</Date>
    <ISBN>94303-12021-43892</ISBN>
    <Publisher>McMillin Publishing</Publisher>
  </Book>...
</BookStore>
```

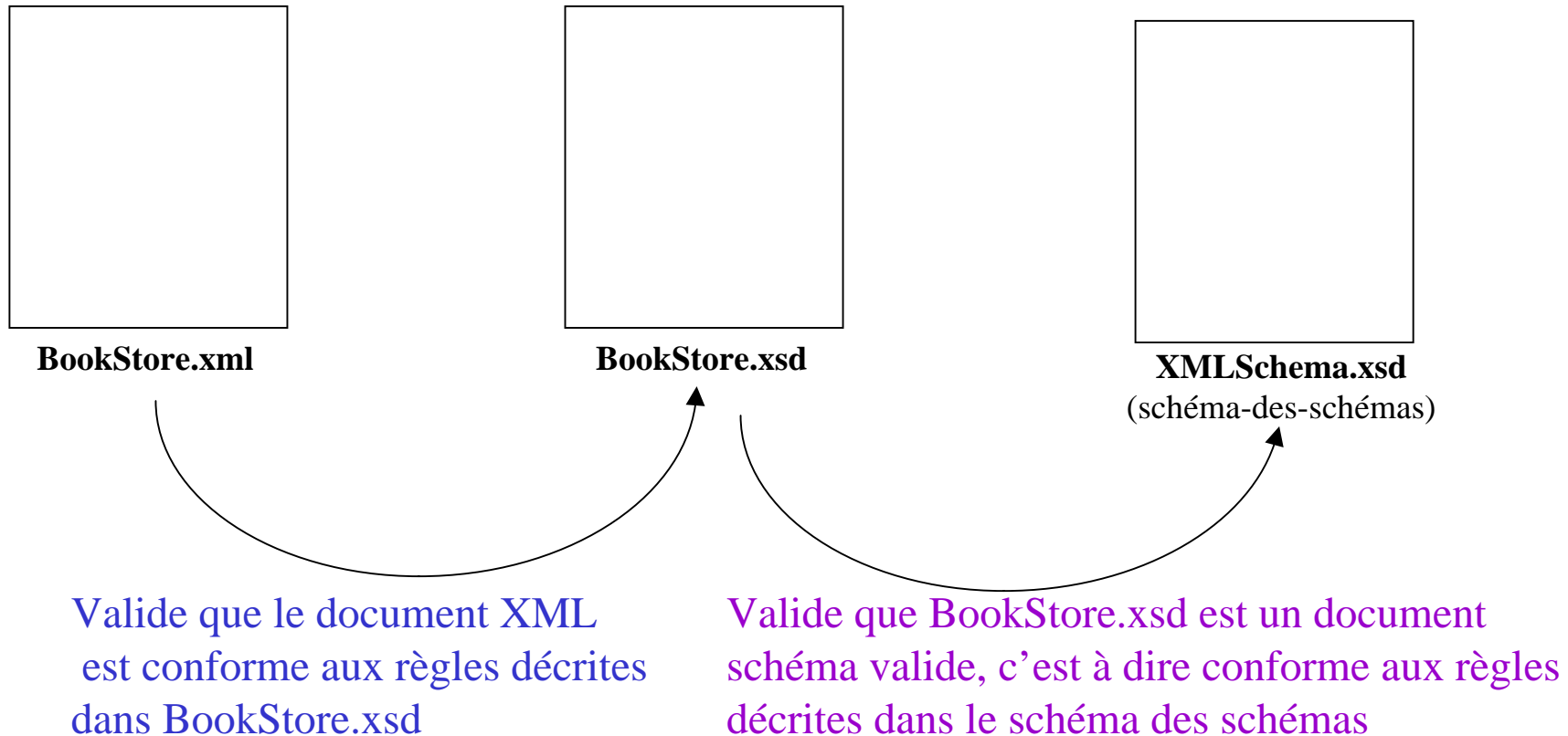
①

②

③

1. Utilisé une déclaration d'espace de noms par défaut indique à l'outil de validation que tous les éléments utilisés dans l'instance document viennent de l'espace de noms *http://www.books.org*
2. *schemaLocation* indique à l'outil de validation que l'espace de noms *http://www.books.org* est défini par *BookStore.xsd* (i.e., ***schemaLocation* contient une paire de valeurs**).
3. Indique à l'outil de validation que l'attribut *schemaLocation* est l'attribut défini dans l'espace de noms *XMLSchema-instance*.

Multiple niveaux de contrôle



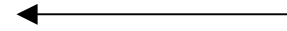
Absence d'espace de noms cible

- Création de schéma sans associer aux éléments un espace de noms
- Ne pas mettre d'attribut *targetNamespace*
- Consequences
 1. Dans le document instance, les éléments ne sont pas qualifiés par un espace de noms
 2. Dans le document instance, au lieu d'utiliser *schemaLocation*
=> Utiliser *noNamespaceSchemaLocation*.

```

<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="BookStore">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Book" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Book">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Title"/>
        <xsd:element ref="Author"/>
        <xsd:element ref="Date"/>
        <xsd:element ref="ISBN"/>
        <xsd:element ref="Publisher"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Title" type="xsd:string"/>
  <xsd:element name="Author" type="xsd:string"/>
  <xsd:element name="Date" type="xsd:string"/>
  <xsd:element name="ISBN" type="xsd:string"/>
  <xsd:element name="Publisher" type="xsd:string"/>
</xsd:schema>

```



Plus d'attribut
targetNamespace,
Plus d'espace de noms
par défaut

Document instance sans espace de noms cible

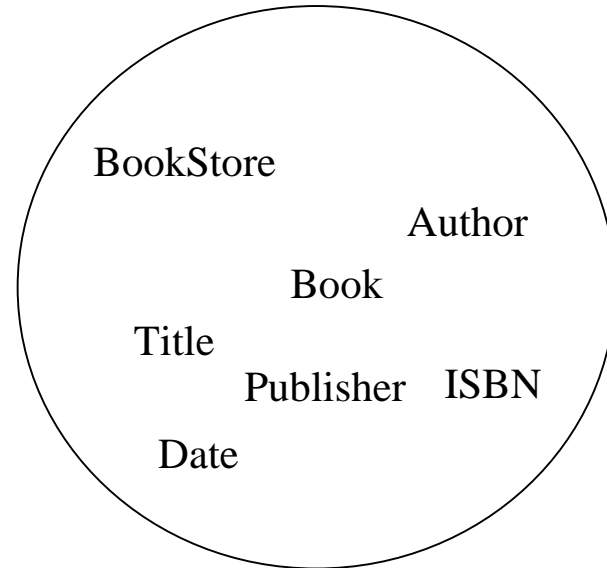
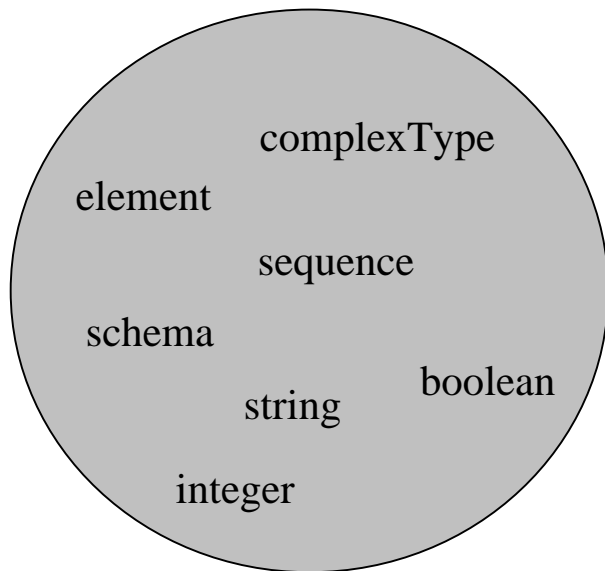
xsi:noNamespaceSchemaLocation est réservé aux références à des schémas décrivant des vocabulaires sans espaces de noms et sa valeur est constituée du chemin permettant d'accéder au schéma correspondant

```
<?xml version="1.0"?>
<BookStore xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation= "BookStore.xsd">
  <Book>
    <Title>Refus de mémoire</Title>
    <Author>Sara Paretsky</Author>
    <Date>2003</Date>
    <ISBN>2-02-081354-5</ISBN>
    <Publisher>Seuil</Publisher>
  </Book>
</BookStore>
```

XMLSchema devient l'espace de noms par défaut

<http://www.w3.org/2001/XMLSchema>

<http://www.books.org> (targetNamespace)



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

```
<schema
```

```
xmlns="http://www.w3.org/2001/XMLSchema"
```

```
targetNamespace="http://www.books.org"
```

```
xmlns:bk="http://www.books.org"
```

```
elementFormDefault="qualified">
```

```
<element name="BookStore">
```

```
<complexType>
```

```
<sequence>
```

```
<element ref="bk:Book" maxOccurs="unbounded"/>
```

```
</sequence>
```

```
</complexType>
```

```
</element>
```

```
<element name="Book">
```

```
<complexType>
```

```
<sequence>
```

```
<element ref="bk:Title"/>
```

```
<element ref="bk:Author"/>
```

```
<element ref="bk:Date"/>
```

```
<element ref="bk:ISBN"/>
```

```
<element ref="bk:Publisher"/>
```

```
</sequence>
```

```
</complexType>
```

```
</element>
```

```
<element name="Title" type="string"/>
```

```
<element name="Author" type="string"/>
```

```
<element name="Date" type="string"/>
```

```
<element name="ISBN" type="string"/>
```

```
<element name="Publisher" type="string"/>
```

```
</schema>
```

Notons que
http://.../XMLSchema
est l'espace de noms
par défaut
En conséquence,
Il n'y a aucune précision
d'espace de noms
sur les éléments

- schema
- element
- complexType
- sequence
- string

Élément d'un schéma

Un élément est défini avec son nom et son type

Syntaxe

```
<xsd:element name="nom" type="nom-type"/>
```

Exemples

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

```
<xsd:element name="age" type="xsd:integer"/>
```

```
<xsd:element name="dateborn" type="xsd:date"/>
```

DTD: <!ELEMENT nom modèle>

```

<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.books.org"
  xmlns="http://www.books.org"
  elementFormDefault="qualified">
  <xsd:element name="BookStore">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Book" maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="Title" type="xsd:string"/>
              <xsd:element name="Author" type="xsd:string"/>
              <xsd:element name="Date" type="xsd:string"/>
              <xsd:element name="ISBN" type="xsd:string"/>
              <xsd:element name="Publisher" type="xsd:string"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>

```

Conception Poupée russe

Il n'y a plus de
référence
à une déclaration
d'éléments

Schéma
plus compact!

Type anonymes

Types nommés

Un type complexe peut être nommé en utilisant l'attribut *type* dans une déclaration d'élément

- Notation

```
<xsd:element name= "nom-element" type= "type-complexe" .../>
```

```
<xsd:complexType name="type-complexe ">
```

- Avantage

Permet de réutiliser ce type dans d'autres éléments

Déclaration d'élément

```
<xsd:element name="name" type="type" minOccurs="int" maxOccurs="int" />
```

↑
↑
↑
Entier positif ou "unbounded"
Entier positif
Un type simple (xsd:string)
ou le nom d'un type complexe (e.g., BookPublication)

```
<xsd:element name="name" minOccurs="int" maxOccurs="int">  
  <xsd:complexType>  
    ...  
  </xsd:complexType>  
</xsd:element>
```

Exemple:BookPublication

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.books.org"
  xmlns="http://www.books.org"
  elementFormDefault="qualified">
  <xsd:element name="BookStore">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Book" type="BookPublication" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:complexType name="BookPublication">
    <xsd:sequence>
      <xsd:element name="Title" type="xsd:string"/>
      <xsd:element name="Author" type="xsd:string"/>
      <xsd:element name="Date" type="xsd:string"/>
      <xsd:element name="ISBN" type="xsd:string"/>
      <xsd:element name="Publisher" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

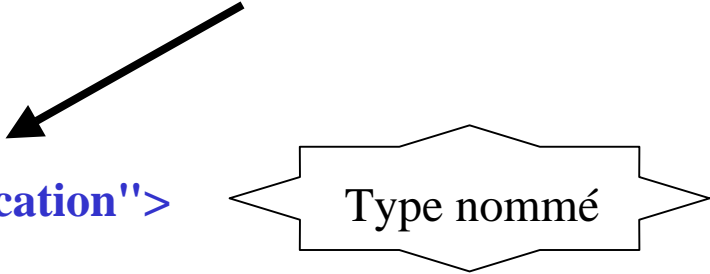
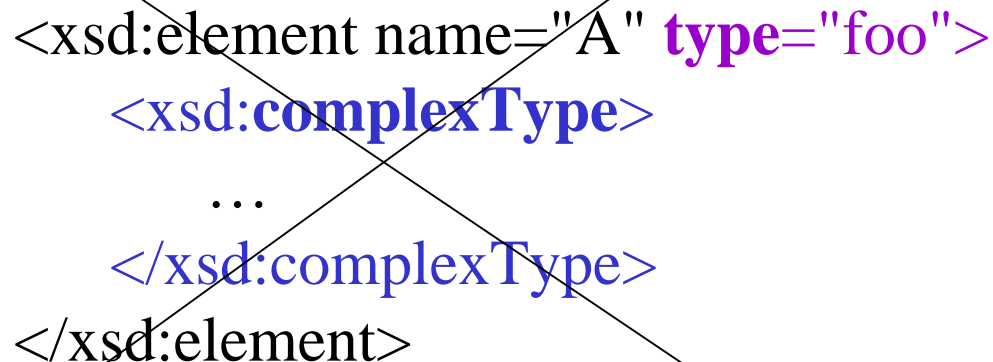


Diagram illustrating the definition of the **BookPublication** type within the **BookStore** element. The **BookPublication** type is defined as a complex type containing a sequence of elements: Title, Author, Date, ISBN, and Publisher, all of type **xsd:string**. The **BookPublication** type is named, as indicated by the callout box labeled "Type nommé".

Attention!!!!

Une déclaration d'élément peut avoir un attribut type ou un type complexe mais pas les deux !



```
<xsd:element name="A" type="foo">  
  <xsd:complexType>  
    ...  
  </xsd:complexType>  
</xsd:element>
```

Déclaration de valeur fixe ou par défaut

Un élément peut avoir une valeur par défaut ou une valeur fixe

Syntaxe

```
<xsd:element name="nom" type="type" default="zzz"/>
```

```
<xsd:element name="nom" type="type" fixed="xxx"/>
```

Exemples

```
<xsd:element name="salary" type="xs:decimal" default="900"/>
```

```
<xsd:element name="color" type="xs:string" fixed="red"/>
```

Problèmes

- Définir l'élément Date comme type string n'est pas satisfaisant

(Cela permet d'affecter n'importe quelle chaîne de caractères)

- Contraindre le contenu de l'élément ISBN
d-ddddd-ddd-d ou d-ddd-ddddd-d ou d-dd-dddddd-d,
où d représente un chiffre

Type de données Date

- Type de donnée prédéfini (i.e., outils de validation connaissent ce type)
- Élément déclaré être de type `date` → `CCYY-MM-DD`
 - intervalle pour CC is: 00-99
 - intervalle pour YY is: 00-99
 - intervalle pour MM is: 01-12
 - intervalle pour DD is:
 - 01-28 si le mois est 2
 - 01-29 si le mois est 2 et `gYear` est une année bissextile
 - 01-30 si le mois est 4, 6, 9, or 11
 - 01-31 si le mois est 1, 3, 5, 7, 8, 10, or 12

Exemple: `1999-05-31` représente `May 31, 1999`

Type de données gYear

- Type de données prédéfini (année du calendrier Gregorian)
- Éléments déclaré de type gYear → CCYY
 - intervalle pour CC is: 00-99
 - intervalle pour YY is: 00-99

Exemple

1999 → gYear 1999

...

```
<xsd:simpleType name="ISBNType">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="\d{1}-\d{5}-\d{3}-\d{1}"/>
    <xsd:pattern value="\d{1}-\d{3}-\d{5}-\d{1}"/>
    <xsd:pattern value="\d{1}-\d{2}-\d{6}-\d{1}"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:element name="BookStore">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Book" maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="Title" type="xsd:string"/>
            <xsd:element name="Author" type="xsd:string"/>
            <xsd:element name="Date" type="xsd:gYear"/>
            <xsd:element name="ISBN" type="ISBNType"/>
            <xsd:element name="Publisher" type="xsd:string"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
</xsd:schema>
```


Nouveau type
ISBNType.

Date est de type gYear,
Et
ISBN de type ISBNType

Expressions équivalentes

```
<xsd:simpleType name="ISBNType">  
  <xsd:restriction base="xsd:string">  
    <xsd:pattern value="\d{1}-\d{5}-\d{3}-\d{1}"/>  
    <xsd:pattern value="\d{1}-\d{3}-\d{5}-\d{1}"/>  
    <xsd:pattern value="\d{1}-\d{2}-\d{6}-\d{1}"/>  
  </xsd:restriction>  
</xsd:simpleType>
```

```
<xsd:simpleType name="ISBNType">  
<xsd:restriction base="xsd:string">  
<xsd:pattern value="\d{1}-\d{5}-\d{3}-\d{1}|\d{1}-\d{3}-\d{5}-\d{1}|\d{1}-\d{2}-\d{6}-\d{1}"/>  
</xsd:restriction>  
</xsd:simpleType>
```



Les barres verticales signifient "ou"

Types prédéfinis

- Type de données primitifs
 - string → "Hello World"
 - boolean → {true, false, 1, 0}
 - decimal → 7.08
 - float → 12.56E3, 12, 12560, 0, -0, INF, -INF, NAN
 - double → 12.56E3, 12, 12560, 0, -0, INF, -INF, NAN
 - duration → P1Y2M3DT10H30M12.3S
 - dateTime → format: CCYY-MM-DDThh:mm:ss
 - time → format: hh:mm:ss.sss
 - date → format: CCYY-MM-DD
 - gYearMonth → format: CCYY-MM
 - gYear → format: CCYY
 - gMonthDay → format: --MM-DD
- Atomic, prédéfini

Notes: 'T' est le séparateur date/temps
INF = infinité
NAN = non un nombre

Types de données primitifs (2)

- Types de données primitifs
 - gDay —————> – format: ---DD (note the 3 dashes)
 - gMonth —————> – format: --MM--
 - hexBinary —————> – a hex string
 - base64Binary —————> – a base64 string
 - anyURI —————> – **http://www.xfront.com**
 - QName —————> – a namespace qualified name
 - NOTATION —————> – a NOTATION from the XML spec
- Atomic, built-in

Types dérivés

- Derived types
 - normalizedString →
 - token →
 - language →
 - IDREFS →
 - ENTITIES →
 - NMTOKEN →
 - NMTOKENS →
 - Name →
 - NCName →
 - ID →
 - IDREF →
 - ENTITY →
 - integer →
 - nonPositiveInteger →
- Subtype of primitive datatype
 - A string without tabs, line feeds, or carriage returns
 - String w/o tabs, l/f, leading/trailing spaces, consecutive spaces
 - any valid xml:lang value, e.g., EN, FR, ...
 - must be used only with attributes
 - must be used only with attributes
 - must be used only with attributes
 - must be used only with attributes
 - **part** (no namespace qualifier)
 - must be used only with attributes
 - must be used only with attributes
 - must be used only with attributes
 - **456**
 - negative infinity to 0

Types dérivés (cont.)

- Derived types
 - negativeInteger → – negative infinity to -1
 - long → – -9223372036854775808 to 9223372036854775807
 - int → – -2147483648 to 2147483647
 - short → – -32768 to 32767
 - byte → – -127 to 128
 - nonNegativeInteger → – 0 to infinity
 - unsignedLong → – 0 to 18446744073709551615
 - unsignedInt → – 0 to 4294967295
 - unsignedShort → – 0 to 65535
 - unsignedByte → – 0 to 255
 - positiveInteger → – 1 to infinity
- Subtype of primitive datatype

<xsd:complexType> ou <xsd:simpleType>?

- Élément ComplexType

Quand des éléments fils doivent être définis et/ou des attributs

- Element simpleType

Quand le nouveau type est juste une restriction de domaine d'un type prédéfini (string, date gYear, etc)

Élément simple

XML élément ne contenant que du texte
⇒ Ne contient ni élément fils ni attributs

- Attention

Peut être de type autre que *string*

- Type prédéfini, dérivé
- Type utilisateur

- Exemples

```
<xsd:element name="lastname" type="xsd:string" />  
<xsd:element name="age" type="xsd:integer" />  
<xsd:element name="hireDate" type="xsd:date" />
```

Définition d'un type de données

Notation

Facets:

- length
- minlength
- maxlength
- pattern
- enumeration
- minInclusive
- maxInclusive
- minExclusive
- maxExclusive
- whiteSpace
- totalDigits
- fractionDigits

```
<xsd:simpleType name= "name">  
  <xsd:restriction base= "xsd:source">  
    <xsd:facet value= "value"/>  
    <xsd:facet value= "value"/>  
    ...  
  </xsd:restriction>  
</xsd:simpleType>
```

Sources:

- string
- boolean
- number
- float
- double
- duration
- dateTime
- time
- ...

Facettes du type de données : string

Le type `string` a six facettes :

- `length`
- `minLength`
- `maxLength`
- `pattern`
- `enumeration`
- `whitespace` (valeurs légales: `preserve`, `replace`, `collapse`)

Exemple:type *phoneNumber*

```
<xsd:simpleType name="phoneNumber"> ①  
  <xsd:restriction base="xsd:string"> ②  
    <xsd:length value="11"/> ③  
    <xsd:pattern value="\d{2}-\d{2}-\d{2}-\d{2}"/> ④  
  </xsd:restriction>  
</xsd:simpleType>
```

1. Définition du type simple *phoneNumber*
2. Éléments de ce type sont des chaînes de caractères
3. De longueur 11
4. La chaîne doit suivre le modèle dd-dd-dd-dd

(la facette *length* est ici redondante par rapport à l'expression du modèle défini par *pattern*)

Expressions régulières sur la facette *pattern*

- La valeur d'une facette *pattern* est une expression régulière de type:

Expression régulière

- Chapter \d
- Chapter \d
- a*b
- [xyz]b
- a?b
- a+b
- [a-c]x

Exemple

- Chapter 1
- Chapter 1
- b, ab, aab, aaab, ...
- xb, yb, zb
- b, ab
- ab, aab, aaab, ...
- ax, bx, cx

Expressions régulières

- Expression régulière
 - [-ac]x
 - [ac-]x
 - [^0-9]x
 - \Dx
 - Chapter\s\d
 - (ho){2} there
 - .abc
 - (a|b)+x
- Exemple
 - -x, ax, cx
 - ax, cx, -x
 - any non-digit char followed by x
 - any non-digit char followed by x
 - Chapter followed by a blank followed by a digit
 - hoho there
 - any (*one*) char followed by abc
 - ax, bx, aax, bbx, abx, bax,...

Expressions régulières

- `\p{L}`
- `\p{Lu}`
- `\p{Ll}`
- `\p{N}`
- `\p{Nd}`
- `\p{P}`
- `\p{Sc}`
- Une lettre de n'importe quel langage,
- Une Lettre majuscule,
- Une lettre minuscule
- Un nombre
- Un chiffre
- Un symbole de ponctuation
- Un signe monétaire

```
<cost>$45.99</cost>  
<cost>¥300</cost>
```

« signe monétaire pour n'importe quel langage suivi par un ou plusieurs chiffres optionnellement suivi par un point et deux chiffres »

Type money

```
<xsd:simpleType name="money">  
  <xsd:restriction base="xsd:string">  
    <xsd:pattern value="\p{Sc}\p{Nd}+(\.\p{Nd}\p{Nd})?" />  
  </xsd:restriction>  
</xsd:simpleType>  
  
<xsd:element name="cost" type="money" />
```

Element lettre

```
<xsd:element name="lettre" >  
  <xsd:simpleType>  
    <xsd:restriction base="xsd:string">  
      <xsd:pattern value="[a-z]"/>  
    </xsd:restriction>  
  </xsd:simpleType>  
</xsd:element>
```

```
<lettre>n</lettre>
```

Exemple: forme

```
<xsd:simpleType name="forme">  
  <xsd:restriction base="xsd:string">  
    <xsd:enumeration value="cercle"/>  
    <xsd:enumeration value="triangle"/>  
    <xsd:enumeration value="rectangle"/>  
  </xsd:restriction>  
</xsd:simpleType>
```

Nouveau type appelé *forme*

Un élément déclaré de type *forme* doit avoir soit la valeur circle, ou triangle, ou rectangle.

Facette whitespace :

preserve : XML processeur respecte les blancs

collapse : suppression de tous les caractères blancs

(line feeds, tabs, spaces, and carriage returns)

replace : tout blanc sera remplacé par un caractère blanc

```
<xsd:element name="address"> <xsd:simpleType>  
  <xsd:restriction base="xsd:string">  
    <xsd:whiteSpace value="preserve"/>  
  </xsd:restriction>  
</xsd:simpleType>  
</xsd:element>
```

Définition d'un type simple à partir d'un type utilisateur simple

```
<xsd:simpleType name= "HauteurImmeuble">  
  <xsd:restriction base="xsd:integer">  
    <xsd:minInclusive value="6"/>  
    <xsd:maxInclusive value="400"/>  
  </xsd:restriction>  
</xsd:simpleType>
```

```
<xsd:simpleType name= " HauteurImmeubleParisien ">  
  <xsd:restriction base=" HauteurImmeuble ">  
    <xsd:minInclusive value="6"/>  
    <xsd:maxInclusive value="120"/>  
  </xsd:restriction>  
</xsd:simpleType>
```

Fixer une valeur de facette

Permet de définir qu'une facette a une valeur invariante.

```
<xsd:simpleType name= "ClassSize">  
  <xsd:restriction base="xsd:nonNegativeInteger">  
    <xsd:minInclusive value="10" fixed="true"/>  
    <xsd:maxInclusive value="60"/>  
  </xsd:restriction>  
</xsd:simpleType>
```

ATTENTION!!!!!!!!!!!!

Les types simples qui dérivent de ce type ne peuvent pas changer cette facette

Type dérivé avec valeur fixe

```
<xsd:simpleType name= "BostonIEEEClassSize">  
  <xsd:restriction base="ClassSize">  
    <xsd:minInclusive value="15"/> ERROR!!!!!!!!!!!!  
    <xsd:maxInclusive value="60"/>  
  </xsd:restriction>  
</xsd:simpleType>
```

Élément contenant un type utilisateur

```
<xsd:simpleType name= "HauteurImmeuble">  
  <xsd:restriction base="xsd:integer">  
    <xsd:minInclusive value="6"/>  
    <xsd:maxInclusive value="400"/>  
  </xsd:restriction>  
</xsd:simpleType>  
  
<xsd:element name= "Hauteur" type=" HauteurImmeuble"/>
```

```
<hauteur>105</hauteur>
```

Déclaration d'éléments

1

```
<xsd:element name="name" type="type" minOccurs="int" maxOccurs="int"/>
```

2

```
<xsd:element name="name" minOccurs="int" maxOccurs="int">  
  ...<xsd:complexType>  
  
    </xsd:complexType>  
</xsd:element>
```

3

```
<xsd:element name="name" minOccurs="int" maxOccurs="int">  
  <xsd:simpleType>  
    <xsd:restriction base="type">  
      ...  
    </xsd:restriction>  
  </xsd:simpleType>  
</xsd:element>
```

Attribut

Un attribut est associé à un type complexe

DTD: `<!ATTLIST nom att typeAtt valeur>`

Syntaxe

`<xsd:attribute name="xxx" type="yyy"/>`

Exemple

`<xsd:attribute name="lang" type="xsd:string"/>`

`<lastname lang="EN"> smith </lastname>`

Propriétés sur les attributs

- Valeur par défaut

```
<xs:attribute name="xxx" type="yyy" default="zzz"/>
```

- valeur fixe

```
<xs:attribute name="xxx" type="yyy" fixed="zzz"/>
```

- attribut optionnel

```
<xs:attribute name="xxx" type="yyy" use="optional"/>
```

- attribut obligatoire

```
<xs:attribute name="xxx" type="yyy" use="required"/>
```


Élément complexe

1. Éléments vide
2. Élément qui comprend d'autres éléments
3. Élément qui comprend uniquement du texte
4. Élément qui comprend du texte et d'autres éléments

- `<product pid="1345"/>`
- `<employee>`
 - `<firstname>John</firstname>`
 - `<lastname>Smith</lastname>``</employee>`
- `<food type="dessert">Ice cream</food>`
- `<letter> Dear Mr.<name>John Smith</name>. Your order <orderid>1032</orderid> will be shipped on <shipdate>2001-07-13</shipdate>. </letter>`

Élément vide

Élément sans contenu mais pouvant admettre des attributs

DTD: `<!ELEMENT product EMPTY>`

`<!ATTLIST product prodid=CDATA #REQUIRED>`

```
<xsd:element name="product">
  <xsd:complexType>
    <xsd:attribute name="prodid"
                  type="xsd:positiveInteger"/>
  </xsd:complexType>
</xsd:element>
```

Élément qui ne comprend que des éléments fils

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

Élément texte

Peut contenir du texte et des attributs

DTD:<!ELEMENT texte (#PCDATA)>

<!ATTLIST texte att=CDATA #IMPLIED>

```
<xsd:element name="elementTexte">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="basetype">
        ....
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
```

Element shoeSize example

```
<xsd:element name="shoeSize" type="shoeType"/>

<xsd:complexType name="shoeType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:integer">
      <xsd:attribute name="country" type="xsd:string" />
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
```

Élément mixte

Un élément mixte peut contenir des attributs
des éléments et du texte.

```
<!ELEMENT nom_type (#PCDATA| ...| nom_fils_n)*>
```

```
<xsd:element name="letter" type="lettertype"/>
```

```
<xsd:complexType name="lettertype" mixed="true">
```

```
<xsd:sequence>
```

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

```
<xsd:element name="orderid" type="xsd:positiveInteger"/>
```

```
<xsd:element name="shipdate" type="xsd:date"/>
```

```
</xsd:sequence>
```

```
</xsd:complexType>
```

Indicateurs de type complexe

Permet de contrôler comment les éléments sont utilisés dans le document

- **Indicateurs d'ordre**
 - All : n'importe quel ordre, une seule occurrence par fils
 - Choice : au choix un des fils
 - Sequence : ordre spécifique des éléments fils
 - **Indicateurs d'occurrence:** fréquence d'occurrence
 - maxOccurs
 - minOccurs
- **Indicateurs de groupe:** ensemble relié d'éléments
 - Group name
 - attributeGroup name

all

n'importe quel ordre, une seule occurrence par fils

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


choice

au choix un des fils

```
<xsd:element name="person">  
  <xsd:complexType>  
    <xsd:choice>  
      <xsd:element name="employee"  
                    type="employee"/>  
      <xsd:element name="member"  
                    type="member"/>  
    </xsd:choice>  
  </xsd:complexType>  
</xsd:element>
```

Sequence

ordre spécifique des éléments fils

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

Indicateurs d'occurrence

- **maxOccurs**

Spécifie le nombre maximum de fois où apparaît un élément

- **minOccurs**

Spécifie le nombre minimum de fois où apparaît un élément

- **remarque**

Pour any, all, choice, sequence, group name, and group reference
la valeur par défaut est de 1 pour ces deux indicateurs

Example

```
<xsd:element name="person">  
  <xsd:complexType>  
    <xsd:sequence>  
      <xsd:element name="full_name"  
                    type="xsd:string"/>  
      <xsd:element name="child_name"  
                    type="xsd:string"  
                    maxOccurs="10"  
                    minOccurs="0"/>  
    </xsd:sequence>  
  </xsd:complexType>  
</xsd:element>
```

Indicateur de groupe

```
<xsd:group name="groupname">
```

...

```
</xsd:group>
```

Groupe nommé "persongroup", qui définit un groupe d'éléments qui doivent arriver dans une séquence exacte

```
<xsd:group name="persongroup">  
  <xsd:sequence>  
    <xsd:element name="firstname" type="xsd:string"/>  
    <xsd:element name="lastname" type="xsd:string"/>  
    <xsd:element name="birthday" type="xsd:date"/>  
  </xsd:sequence>  
</xsd:group>
```

Utilisation d'un groupe

Un groupe peut être utilisé dans un autre type d'élément

```
<xsd:element name="person" type="personinfo"/>

<xsd:complexType name="personinfo">
  <xsd:sequence>
    <xsd:group ref="persongroup"/>
    <xsd:element name="country"
                  type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
```

Attribut group

Permet de regrouper un ensemble d'attributs

```
<xsd:attributeGroup name="personattrgroup">
  <xsd:attribute name="firstname" type="xs:string"/>
  <xsd:attribute name="lastname" type="xs:string"/>
  <xsd:attribute name="birthday" type="xs:date"/>
</xsd:attributeGroup>

<xsd:element name="person">
  <xsd:complexType>
    <xsd:attributeGroup ref="personattrgroup"/>
  </xsd:complexType>
</xsd:element>
```

Element Any

Permet d'étendre un document XML avec des éléments non spécifié

```
<xsd:element name="person">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="firstname" type="xs:string"/>
      <xsd:element name="lastname" type="xs:string"/>
      <xsd:any minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```


Attribut Any

Permet d'avoir dans un document instance des attributs non spécifiés dans son schéma

```
<xsd:element name="Book">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Title" type="xsd:string"/>
      <xsd:element name="Author" type="xsd:string"/>
      <xsd:element name="Date" type="xsd:string"/>
      <xsd:element name="ISBN" type="xsd:string"/>
      <xsd:element name="Publisher" type="xsd:string"/>
      <xsd:any minOccurs="0"/>
    </xsd:sequence>
    <xsd:anyAttribute/>
  </xsd:complexType>
</xsd:element>
```

Group substitution

Permet de définir un groupe d'éléments substituables

- Deux parties
 - Élément substituable
 - Éléments qui le remplace

```
<xsd:element name="subway" type="xsd:string"/>  
<xsd:element name="T" substitutionGroup="subway" type="xsd:string"/>
```

subway est l'élément substituable

T est substituable pour **subway**

Example

```
<xsd:element name="subway" type="xsd:string"/>
  <xsd:element name="metro" substitutionGroup="subway"
    type="xsd:string"/>
<xsd:element name="transportation" type = "TypeTransportation "/>
  <xsd:complexType name= "TypeTransportation">
    <xsd:sequence>
      <xsd:element ref="subway"/>
    </xsd:sequence>
  </xsd:complexType>

<xsd:element name= "transport" substitutionGroup="transportation" />
```

Document initial

en Français transport remplace transportation
metro remplace subway

```
<transportation>
  <subway>Red Line</subway>
</transportation>
```

```
<transport>
  <metro>Porte Dauphine</metro>
</transport>
```

Blocage de la substitution

Un élément interdit la substitution avec un autre élément en ajoutant l'attribut *block*

```
<xsd:element name="name_el" type="type_el" block="substitution"/>
```

Exemple

```
<xsd:element name="name" type="xsd:string" block="substitution" />  
<xsd:element name="nom" substitutionGroup="name"/> ERREUR!!!!
```

Remarque substitutionGroup

Le type de chaque élément dans un groupe de substitution doit être le même, ou dérivé du type à substituer.

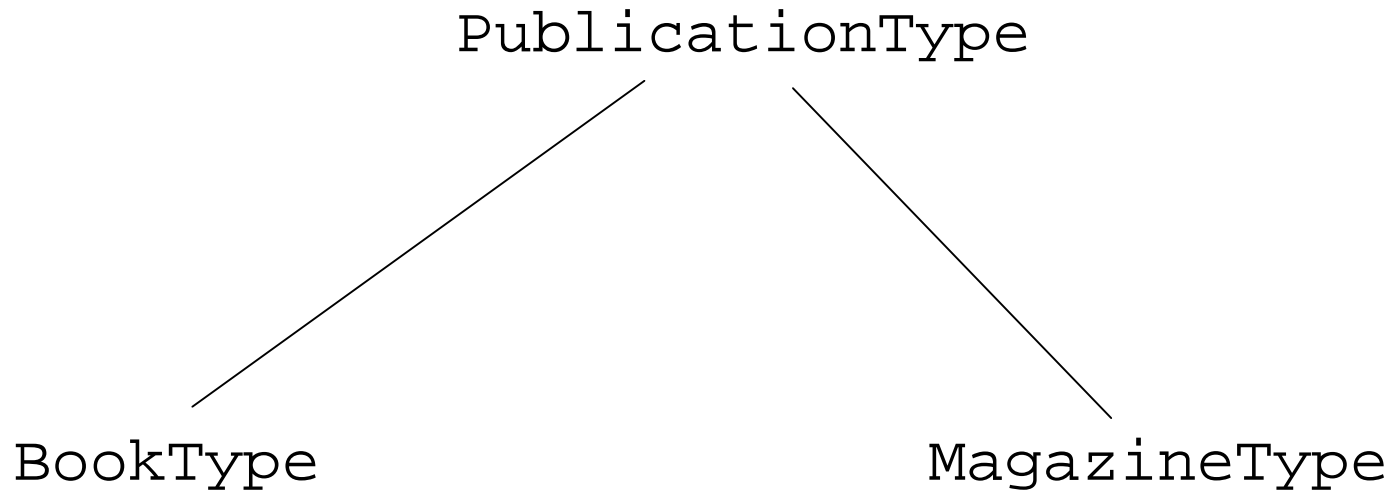
```
<xsd:element name="A" type="TypeA"/>
```

```
<xsd:element name="B" substitutionGroup="A" type="yyy"/>
```



type yyy doit être soit TypeA
soit un dérivé de TypeA

BookType et MagazineType Dérivent de PublicationType



Élément substitution avec les types dérivés

```
<xsd:element name="Publication" type="PublicationType"/>
<xsd:element name="Book" substitutionGroup="Publication"
               type="BookType"/>
<xsd:element name="Magazine" substitutionGroup="Publication"
               type="MagazineType"/>
<xsd:element name="BookStore">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="Publication" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

```

<xsd:complexType name="PublicationType">
  <xsd:sequence>
    <xsd:element name="Title" type="xsd:string"/>
    <xsd:element name="Author" type="xsd:string" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="Date" type="xsd:gYear"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="BookType">
  <xsd:complexContent>
    <xsd:extension base="PublicationType" >
      <xsd:sequence>
        <xsd:element name="ISBN" type="xsd:string"/>
        <xsd:element name="Publisher" type="xsd:string"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="MagazineType">
  <xsd:complexContent>
    <xsd:restriction base="PublicationType">
      <xsd:sequence>
        <xsd:element name="Title" type="xsd:string"/>
        <xsd:element name="Date" type="xsd:gYear"/>
      </xsd:sequence>
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>

```



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

```
<BookStore>
```

```
  <Book>
```

```
    <Title>Persuasion</Title>
```

```
    <Author>Jane Austen</Author>
```

```
    <Date>1967</Date>
```

```
    <ISBN>0-441-34319-4</ISBN>
```

```
    <Publisher>Gallimard</Publisher>
```

```
  </Book>
```

```
  <Magazine>
```

```
    <Title>Magic Revue Pop Moderne</Title>
```

```
    <Date>2006</Date>
```

```
  </Magazine>
```

```
  <Book>
```

```
    <Title>Meurtre sur la route de Bethléem</Title>
```

```
    <Author>Batya Gour</Author>
```

```
    <Date>2003</Date>
```

```
    <ISBN>2-07-030898-7</ISBN>
```

```
    <Publisher>Gallimard Foliopolicier</Publisher>
```

```
  </Book>
```

```
</BookStore>
```

<BookStore> peut
contenir des éléments
qui se sont substitués
à Publication!