# W3C XML Schema

Philippe.Poulard@inria.fr

# Sommaire

Les schémas décrivent la grammaire utilisable dans un document XML
Ils décrivent la structure et permettent de définir des types de données

<table>
<tr><td>Document Type Definition DTD</td><td>syntaxe non XML</td><td rowspan="2">W3C</td></tr>
<tr><td>W3C XML Schema WXS</td><td rowspan="2">syntaxe XML</td></tr>
<tr><td>Relax-NG</td><td>ISO</td></tr>
</table>

Spécifications W3C :

- W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures

  `http://www.w3.org/TR/xmlschema11-1/`

- W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes

  `http://www.w3.org/TR/xmlschema11-2/`

Schema du W3C XML Schema `http://www.w3.org/2012/04/XMLSchema.xsd`

# Exemple



"shiporder.xsd"

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="shiporder">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="orderperson" type="xs:string"/>
        <xs:element name="shipto">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="name" type="xs:string"/>
              <xs:element name="address" type="xs:string"/>
              <xs:element name="city" type="xs:string"/>
              <xs:element name="country" type="xs:string"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="item" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="title" type="xs:string"/>
              <xs:element name="note" type="xs:string" minOccurs="0"/>
              <xs:element name="quantity" type="xs:positiveInteger"/>
              <xs:element name="price" type="xs:decimal"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="orderid" type="xs:integer" use="required"/>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

Deux espaces de noms sont définis dans la spécification :

Celui utilisé dans les schémas (associé la plupart du temps au préfixe `xs`)

```
http://www.w3.org/2001/XMLSchema
```

Celui utilisé dans les instances XML (associé la plupart du temps au préfixe `xsi`)

```
http://www.w3.org/2001/XMLSchema-instance
```

**Attributs d'instance**

```
xsi:type
xsi:nil
xsi:schemaLocation
xsi:noNamespaceSchemaLocation
```

# Référencement sans espace de nom



```
<?xml version="1.0" encoding="UTF-8"?>
<shiporder orderid="889923"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="shiporder.xsd">
  <orderperson>John Smith</orderperson>
  <shipto>
    <name>Ola Nordmann</name>
    <address>Langgt 23</address>
    <city>4000 Stavanger</city>
    <country>Norway</country>
  </shipto>
  <item>
    <title>Empire Burlesque</title>
    <note>Special Edition</note>
    <quantity>1</quantity>
    <price>10.90</price>
  </item>
  <item>
    <title>Hide your heart</title>
    <quantity>1</quantity>
    <price>9.90</price>
  </item>
</shiporder>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE shiporder SYSTEM "shiporder.dtd">
<shiporder orderid="889923">
  <orderperson>John Smith</orderperson>
  <shipto>
    <name>Ola Nordmann</name>
    <address>Langgt 23</address>
    <city>4000 Stavanger</city>
```

# Référencement avec espace de nom

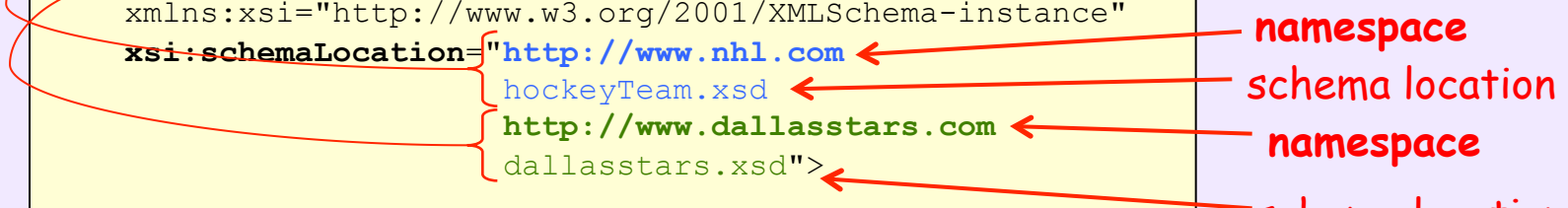


- Un schéma par espace de nom
- Une paire de valeurs {namespace, localisation} par schéma

```
<?xml version="1.0"?>
<hockeyTeam
    xmlns="http://www.nhl.com"
    xmlns:stars="http://www.dallasstars.com"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.nhl.com
                        hockeyTeam.xsd
                        http://www.dallasstars.com
                        dallasstars.xsd">

    <team name="Dallas Stars"
          stars:logo="http://www.dallasstars.com/logo.jpg">
                    <!-- More XML Content -->
    </team>

</hockeyTeam>
```



# Target namespace



- Les noms **globaux** (noms d'élément, noms de type, et noms d'attributs), c'est à dire définis immédiatement dans `<xsd:schema>` appartiennent au *target namespace*.
- Les noms d'élément **locaux** appartiennent au *target namespace* si `elementFormDefault="qualified"`
- Les noms d'attributs **locaux** appartiennent au *target namespace* si `attributeFormDefault="qualified"`

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:tns="http://www.example.com/store"
        targetNamespace="http://www.example.com/store"
        attributeFormDefault="unqualified"
        elementFormDefault="qualified">

  <xsd:element name="InvoiceNo">
     <xsd:complexType>
        <xsd:attribute name="orderid" use="required"
                       type="xsd:positiveInteger"/>
     </xsd:complexType>
  </xsd:element>

  <xsd:element name="ProductID" type="tns:ProductCode"/>

  <xsd:simpleType name="ProductCode">
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="[A-Z]{1}d{6}"/>
    </xsd:restriction>
  </xsd:simpleType>

</xsd:schema>
```

Par défaut, la valeur est "unqualified", donc les attributs **locaux** ne sont pas définis dans le *target namespace*

Les références aux types sont des QNames, donc ils peuvent avoir un préfixe

Le *target namespace* est "`http://www.example.com/store`", il contient les noms :

- **`InvoiceNo`** (nom global), **`ProductID`** (nom global),
- et **`ProductCode`** (nom global),
- mais pas **`orderid`** car c'est un attribut **local**, que la directive `attributeFormDefault="unqualified"` exclue du *target namespace*.

# Target namespace



```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns    ="http://www.example.com/store"
      targetNamespace="http://www.example.com/store"
      attributeFormDefault="unqualified"
      elementFormDefault="qualified">

  <xsd:element name="InvoiceNo">
     <xsd:complexType>
        <xsd:attribute name="orderid" use="required"
                       type="xsd:positiveInteger"/>
     </xsd:complexType>
  </xsd:element>

  <xsd:element name="ProductID" type="ProductCode"/>

  <xsd:simpleType name="ProductCode">
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="[A-Z]{1}d{6}"/>
    </xsd:restriction>
  </xsd:simpleType>

</xsd:schema>
```

# Target namespace



```
<schema      xmlns     ="http://www.w3.org/2001/XMLSchema"
             xmlns:tns="http://www.example.com/store"
       targetNamespace="http://www.example.com/store"
       attributeFormDefault="unqualified"
       elementFormDefault="qualified">

  <element      name="InvoiceNo">
     <complexType>
        <attribute      name="orderid" use="required"
                        type="positiveInteger"/>
     </complexType>
  </element>

  <element      name="ProductID" type="tns:ProductCode"/>

  <simpleType      name="ProductCode">
    <restriction      base="string">
      <pattern      value="[A-Z]{1}d{6}"/>
    </restriction>
  </simpleType>

</schema>
```

# <xs:element>

Eléments parent :

- <xs:schema>,
- <xs:choice>,
- <xs:all>,
- <xs:sequence>,
- <xs:group>.

```
<xs:element
    abstract = boolean : false
    block = (#all | List of (extension | restriction | substitution))
    default = string
    final = (#all | List of (extension | restriction))
    fixed = string
    form = (qualified | unqualified)
    id = ID
    maxOccurs = (nonNegativeInteger | unbounded)  : 1
    minOccurs = nonNegativeInteger : 1
    name = NCName
    nillable = boolean : false
    ref = QName
    substitutionGroup = List of QName
    targetNamespace = anyURI
    type = QName
    {any attributes with non-schema namespace . . .}>

  Content: (<xs:annotation>?, ((<xs:simpleType> | <xs:complexType>)?,
                   <xs:alternative>*, (<xs:unique> | <xs:key> | <xs:keyref>)*))

</xs:element>
```

XML :

```
<employee>
  <firstName>John</firstName>
  <lastName>Smith</lastName>
</employee>
```

## Référence à des éléments globaux

```
<xs:element name="employee">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="firstName"/>
      <xs:element ref="lastName"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="firstName" type="xs:string"/>
<xs:element name="lastName" type="xs:string"/>
```

## Utilisation d'éléments locaux

```
<xs:element name="employee">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstName" type="xs:string"/>
      <xs:element name="lastName" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Les éléments et attributs doivent avoir un type :

- `xs:simpleType` : pour les d'attributs
  et les éléments ne contenant que du texte

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

- `xs:complexType` : pour les éléments :
  - ayant des attributs
  - comportant d'autres éléments, ou aucun
  - comportant du contenu mixte

Un type peut être :

Anonyme : défini dans la déclaration de l'élément

```
<xs:element name="employee">
    <xs:complexType>
      <xs:attribute name="lastName" type="xs:string" use="required"/>
      <xs:attribute name="firstName" type="xs:string" use="required"/>
    </xs:complexType>
</xs:element>
```

Type anonyme (local)

Nommé : déclaré globalement et référencé par l'élélément

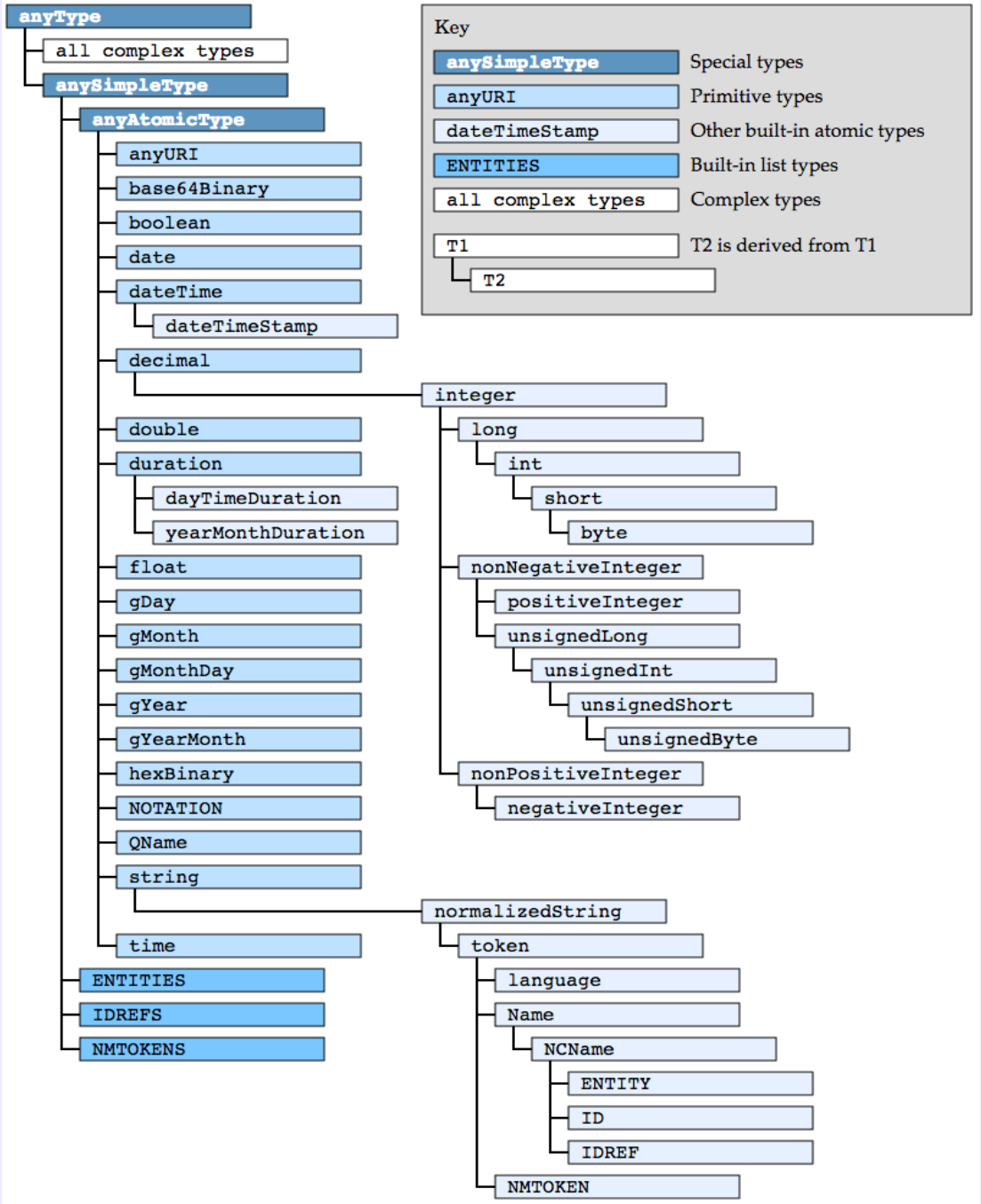
```
<xs:complexType name="employeeType">
  <xs:attribute name="lastName" type="xs:string" use="required"/>
  <xs:attribute name="firstName" type="xs:string" use="required"/>
</xs:complexType>

<xs:element name="employee" type="employeeType"/>
```

Type nommé (global)

# Le système de typage





Contrainte sur la valeur terminale d'un élément ou d'un attribut

Un type de données dispose de 3 propriétés :

- Un espace de valeurs
- Un espace lexical qui correspond aux littéraux utilisés dans le document XML
- Des fonctionnalités, relations et procédures associées
  (égalité, relation d'ordre, etc)

**Exemple** : les littéraux `'0.1'` et `'0.10000000009'` pour le type `xs:float` n'ont pas de valeur correspondantes dans l'espace de valeur, et sont mappées sur la même valeur (la plus proche) : `0.100000001490116119384765625`

Un type simple peut être dérivé :

- Par **restriction** : l'élément `<xs:restriction>` restreint la plage de valeurs (de l'espace de valeur ou de l'espace lexical) pour le type simple à un sous-ensemble de celles du type simple hérité.
  Les restrictions s'appellent les *facettes*.

- Par **liste** : l'élément `<xs:list>` définit un type simple qui contient une liste, séparée par des espaces blancs, de valeurs d'un type simple hérité.

- Par **union** : l'élément `<xs:union>` définit un type simple qui contient une union des valeurs de plusieurs types simples hérités.

Contraintes possibles pour la restriction :

| *Constraint* | *Description* |
|---|---|
| enumeration | Defines a list of acceptable values |
| fractionDigits | Specifies the maximum number of decimal places allowed.<br>Must be equal to or greater than zero |
| length | Specifies the exact number of characters or list items allowed.<br>Must be equal to or greater than zero |
| maxExclusive | Specifies the upper bounds for numeric values (the value must be less than this value) |
| maxInclusive | Specifies the upper bounds for numeric values<br>(the value must be less than or equal to this value) |
| maxLength | Specifies the maximum number of characters or list items allowed.<br>Must be equal to or greater than zero |
| minExclusive | Specifies the lower bounds for numeric values<br>(the value must be greater than this value) |
| minInclusive | Specifies the lower bounds for numeric values<br>(the value must be greater than or equal to this value) |
| minLength | Specifies the minimum number of characters or list items allowed.<br>Must be equal to or greater than zero |
| pattern | Defines the exact sequence of characters that are acceptable |
| totalDigits | Specifies the exact number of digits allowed. Must be greater than zero |
| whiteSpace | Specifies how white space (line feeds, tabs, spaces, and carriage returns) is handled |

## <xs:enumeration>

```
<xs:simpleType name="WayType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="street"/>
    <xs:enumeration value="road"/>
    <xs:enumeration value="avenue"/>
    <xs:enumeration value="boulevard"/>
  </xs:restriction>
</xs:simpleType>
```

## Facettes

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

<xs:whitespace> :

- preserve : les espaces sont conservés
- replace : les tabulations et retours chariots sont remplacés par des espaces
- collapse : les séquences de tabulations et retours chariots sont remplacés par un seul espace ; les espaces de début et de fin sont supprimés

```
<xs:element name="age">
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="120"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

```
<xs:element name="password">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:minLength value="5"/>
      <xs:maxLength value="8"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

## <xs:pattern>

Pour contraindre l'espace lexical avec une expression régulière

```
<xs:simpleType name="OrderIdCode">
   <xs:restriction base="xs:string">
     <xs:pattern value="[A-Z][0-9A-Z]*"/>
   </xs:restriction>
 </xs:simpleType>
```

Les expressions régulières sont dépourvues d'ancres (signes `^` et `$`) car elles sont implicites : une contrainte est susceptible de s'appliquer sur toutes les données.

Le pattern `regexp` est équivalent à `^regexp$` dans les autres validateurs d'expression.

Pour rechercher un pattern dans un texte, utiliser `.*regexp.*`,
ou `[\s\S]*regex[\s\S]*` pour inclure les retours à la ligne

# <xs:list>

Définit une collection d'items du même type

```
<xs:simpleType name='listOfIntegers'>
    <xs:list itemType='xs:integer'/>
</xs:simpleType>
```

Lorsqu'un type de données est dérivé d'un type liste, les facettes `length`, `maxLength` et `minLength` portent sur le nombre d'items de la liste

```
<xs:simpleType name='fiveIntegers'>
  <xs:restriction base='listOfIntegers'>
    <xs:maxLength value='5'>
  </xs:restriction>
</xs:simpleType>

<xs:element name="suite" type="fiveIntegers"/>
```

XML :

```
<suite>1 100 9 4000 0</suite>
```

# <xs:union>

Définit une collection de types multiples

```
<xs:element name="fontsize">
    <xs:simpleType>
        <xs:union memberTypes="fontNumber fontName" />
    </xs:simpleType>
</xs:element>

<xs:simpleType name="fontNumber">
    <xs:restriction base="xs:positiveInteger">
         <xs:maxInclusive value="110"/>
    </xs:restriction>
</xs:simpleType>

<xs:simpleType name="fontName">
    <xs:restriction base="xs:string">
        <xs:enumeration value="small"/>
        <xs:enumeration value="medium"/>
        <xs:enumeration value="large"/>
    </xs:restriction>
</xs:simpleType>
```

XML :

```
<fontsize>24</fontsize>
<fontsize>small</fontsize>
<fontsize>48</fontsize>
<fontsize>large</fontsize>
```

# Types complexes <xs:complexType>



Concerne :

- Les éléments vides
- Les éléments qui contiennent d'autres éléments
- Les éléments qui contiennent des attributs
- Les éléments qui contiennent du contenu mixte

Les types permettent la réutilisation d'un même modèle de contenu, et peuvent être étendus

```
<xs:element name="student" type="personinfo"/>
<xs:element name="member" type="personinfo"/>
<xs:element name="employee" type="fullpersoninfo"/>

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

<xs:complexType name="fullpersoninfo">
  <xs:complexContent>
    <xs:extension base="personinfo">
      <xs:sequence>
        <xs:element name="address" type="xs:string"/>
        <xs:element name="city" type="xs:string"/>
        <xs:element name="country" type="xs:string"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Les modèles de contenus comportent les attributs et la définition des sous-éléments.

Les sous-élément peuvent être connectés par :

- <xs:sequence> : liste ordonnée
- <xs:choice> : choix
- <xs:all> : liste non-ordonnée (tous les sous-elements sont optionels)

## <xs:sequence>

Exige que les éléments du groupe s'affichent dans l'ordre spécifié dans l'élément conteneur.

```
<xs:complexType name="DirectoryType">
  <xs:sequence minOccurs="1" maxOccurs="unbounded">
    <xs:element name="people" type="PeopleType" minOccurs="1" maxOccurs="1"/>
    <xs:element name="address" type="AddressType" minOccurs="1" maxOccurs="1"/>
  </xs:sequence>
  <xs:attribute name="name" type="xs:string" use="optional" />
</xs:complexType>

<xs:element name="directory" type="DirectoryType"/>
```

- Les attributs d'occurrence `minOccurs` et `maxOccurs` contraignent le nombre de fois qu'un contenu doit se répéter.
- La valeur `unbounded` permet de ne pas imposer de limite supérieure
- La valeur par défaut de chaque attribut est 1.

XML :

```
<directory>
    <people …>…</people>
    <address>…</address>
    <people …>…</people>
    <address>…</address>
    <people …>…</people>
    <address>…</address>
</directory>
```

# <xs:choice>

Autorise un et un seul des éléments contenus dans le groupe sélectionné à être présent dans l'élément conteneur.

```
<xs:complexType name="sectionType">
  <xs:choice maxOccurs="unbounded">
    <xs:element name="par" type="parType" />
    <xs:element name="image" type="imageType" />
    <xs:element name="ol" type="listType" />
    <xs:element name="ul" type="listType" />
  </xs:choice>
</xs:complexType>

<xs:element name="section" type="sectionType"/>
```

XML :

```
<section>
    <par>…</par>
    <image></image>
    <par>…</par>
    <par>…</par>
    <ol>…</ol>
</section>
```

# <xs:extension>

```
<xs:complexType name="address">
  <xs:sequence>
    <xs:element name="street" type="xs:string" />
    <xs:element name="city" type="xs:string" />
  </xs:sequence>
</xs:complexType>

<xs:complexType name="USAddress">
  <xs:complexContent>
    <xs:extension base="address">
      <xs:sequence>
        <xs:element name="state" type="xs:string" />
      </xs:sequence>
      <xs:attribute name="country" type="xs:string" fixed="US" />
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

XML :

```
<USAddress country="US">
    <street>Pine street</street>
    <city>Syracuse</city>
    <state>New York</state>
</USAddress>
```

## <xs:group>

```
<xs:group name="paragraphs-etc">
  <xs:choice>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element name="par" type="par-type" />
      <xs:element name="image" type="image-type" />
      <xs:element name="ol" type="list-type" />
      <xs:element name="ul" type="list-type" />
    </xs:choice>
  </xs:choice>
</xs:group>

<xs:complexType name="chapter-type">
  <xs:choice maxOccurs="unbounded">
    <xs:element name="section" type="section-type" />
    <xs:group ref="paragraphs-etc"/>
  </xs:choice>
</xs:complexType>

<xs:complexType name="section-type">
  <xs:group ref="paragraphs-etc"/>
</xs:complexType>
```

# Groupe de substitution



- Permet de définir une collection d'éléments à partir d'un élément générique
- L'élément remplaçable est appelé la tête, et doit être définit globalement
- Les éléments du groupe de substitution doivent être du même type que la tête ou un type dérivé

```
<xs:element name="widget" type="xsd:string" />
<xs:element name="woodWidget" type="xsd:string" substitutionGroup="widget" />
```

## xsi:type

- Le type d'un élément peut être surchargé en positionnant la valeur de l'attribut `xsi:type`
- Le nouveau type doit être dérivé du type surchargé

```
<root xsi:noNamespaceSchemaLocation="exemple.xsd"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
  <e>123.4</e>
</root>
```

```
<root xsi:noNamespaceSchemaLocation="exemple.xsd"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema" >
  <e xsi:type="xsd:integer">10</e>
</root>
```

# xsi:nil

- Notion de "null" dans XML
- `xsi:nil` indique qu'un élément doit être accepté comme valide quand il n'a pas de contenu, bien que son modèle de contenu interdise qu'il soit vide
- Un tel élément doit alors être vide, mais peut porter des attributs

```
<xs:element name="author" nillable="true">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="name"/>
      <xs:element ref="born"/>
      <xs:element ref="dead" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute ref="id"/>
  </xs:complexType>
</xs:element>
```

XML :

```
<author xsi:nil="true"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
```

# <xs:attribute>

Eléments parent :

- <xs:attributeGroup>,
- <xs:schema>,
- <xs:complexType>,
- <xs:restriction>,
- <xs:extension>.

```
<xs:attribute
  default = string
  fixed = string
  form = (qualified | unqualified)
  id = ID
  name = NCName
  ref = QName
  targetNamespace = anyURI
  type = QName
  use = (optional | prohibited | required) : optional
  inheritable = boolean
  {any attributes with non-schema namespace . . .}>

    Content: (<xs:annotation>?, <xs:simpleType>?)

</xs:attribute>
```

**use="optional | required | prohibited"**

```
<xs:attribute name="seatType" type="xs:string"/>

<xs:attribute name="seatType" type="xs:string" use="optional"/>
```

```
<xs:complexType name="A">
  <xs:attribute name="x" type="xs:string"/>
  <xs:attribute name="y" type="xs:positiveInteger"/>
 </xs:complexType>

<xs:complexType name="B">
  <xs:complexContent>
   <xs:restriction base="A">
    <xs:attribute name="x" use="required" />
    <xs:attribute name="y" use="prohibited"/>
   </xs:restriction>
  </xs:complexContent>
</xs:complexType>
```

<xs:import> permet d'utiliser plusieurs schémas ayant des espaces de nommage (*target namespace*) différents

```
<xs:import
          id=ID
          namespace=anyURI
          schemaLocation=anyURI
          any attributes>

          Content: (<xs:annotation>?)

</xs:import>
```

- L'attribut `namespace` référence l'URI de l'espace de noms à importer.
  Si cet attribut est absent, le schéma conteneur peut contenir des références non qualifiées à des composants figurant dans l'espace de noms importé.

- L'attribut `schemaLocation` référence l'URI d'emplacement du schéma pour l'espace de noms importé. S'il est absent, l'URI est spécifiée par le document XML.

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

    <xs:import namespace="http://www.example.fr/schema"/>
    ...
</xs:schema>
```

<xs:include> permet d'utiliser plusieurs schémas ayant le même espace de nommage (*target namespace*)

```
<xs:include
          id=ID
          schemaLocation=anyURI
          any attributes>

          Content: (<xs:annotation>?)

</xs:include>
```

- L'attribut `schemaLocation` référence l'URI d'emplacement du schéma pour l'espace de noms à inclure. Il est obligatoire.

```
<?xml version="1.0"?>
<xs:schema          xmlns:xs="http://www.w3.org/2001/XMLSchema"
                    targetNamespace="http://www.example.com/schema">
    <xs:include schemaLocation="http://www.example.com/schema/course.xsd"/>
    <xs:include schemaLocation="http://www.example.com/schema/training.xsd"/>
    ...
</xs:schema>
```

# <xs:unique>



<xs:unique> spécifie qu'une valeur (ou une combinaison de valeurs) d'attribut ou d'élément doit être unique dans la portée spécifiée. La valeur doit être unique ou null.

```
<xs:unique
          id=ID
          name=NCName
          any attributes>

          Content: (<xs:annotation>?,(<xs:selector>, <xs:field>+))

</xs:unique>
```

- L'élément **<xs:selector>** contient une expression XPath (XML Path language) spécifiant l'ensemble d'éléments dans lequel les valeurs spécifiées par les éléments **<xs:field>** doivent être uniques.
- Chaque élément **<xs:field>** contient une expression XPath spécifiant les valeurs (d'attribut ou d'élément) qui doivent être uniques pour l'ensemble d'éléments spécifiés par l'élément **<xs:selector>**.
- En présence de plusieurs éléments **<xs:field>**, leur combinaison doit être unique. Dans ce cas, les valeurs d'un élément **<xs:field>** unique peuvent ou non être uniques pour les éléments sélectionnés, mais la combinaison de tous les champs doit être unique.

```
<xs:simpleType name="USERS">
    <xs:restriction base="xs:string">
        <xs:enumeration value="Root"/>
        <xs:enumeration value="Admin"/>
        <xs:enumeration value="User"/>
        <xs:enumeration value="Guest"/>
    </xs:restriction>
</xs:simpleType>

<xs:element name="permission">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="userType" type="USERS"
                    minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <xs:unique name="uniqueUserType">
        <xs:selector xpath="userType"/>
        <xs:field xpath="."/>
    </xs:unique>
</xs:element>
```

XML :

```
<permission>
     <userType>Root</userType>
     <userType>Admin</userType>
 </permission>
```

<xs:key> spécifie qu'un attribut ou qu'une valeur (ou ensemble de valeurs) d'élément doit être une clé dans la portée spécifiée. La portée d'une clé est l'élément <xs:element> conteneur dans un document d'instance. Une clé signifie que les données doivent être uniques dans une portée spécifiée, qui ne peuvent prendre de valeur null et toujours présentes.

```
<xs:key   id=ID
          name=NCName
          any attributes>

          Content: (<xs:annotation>?,(<xs:selector>, <xs:field>+))

</xs:key>
```

- L'élément <xs:selector> contient une expression XPath spécifiant l'ensemble d'éléments dans lequel les valeurs spécifiées par <xs:field> doivent être uniques.
- Chaque élément <xs:field> contient une expression XPath spécifiant les valeurs (d'attribut ou d'élément) qui doivent être uniques pour l'ensemble d'éléments spécifiés par l'élément <xs:selector>.
- En présence de plusieurs éléments <xs:field>, leur combinaison doit être unique. Dans ce cas, les valeurs d'un élément <xs:field> unique peuvent ou non être uniques pour les éléments sélectionnés, mais la combinaison de tous les champs doit être unique.

# Référence à une clé



<xs:keyref> spécifie qu'une valeur (ou un ensemble de valeurs) d'attribut ou d'élément correspond à celle de l'élément <xs:key> ou <xs:unique> spécifié.

```
<xs:keyref
          id=ID
          name=NCName
          refer=QName
          any attributes>

     Content: (<xs:annotation>?,(<xs:selector>,<xs:field>+))

</xs:keyref>
```

- L'attribut `refer` contient le nom d'un élément <xs:key> ou <xs:unique>.
- L'élément <xs:selector> contient une expression XPath spécifiant l'ensemble d'éléments dans lequel les valeurs spécifiées par <xs:field> doivent être uniques.
- Chaque élément <xs:field> contient une expression XPath spécifiant les valeurs (d'attribut ou d'élément) qui doivent être uniques pour l'ensemble d'éléments spécifiés par l'élément <xs:selector>.
- En présence de plusieurs éléments <xs:field>, leur combinaison doit être unique. Dans ce cas, les valeurs d'un élément <xs:field> unique peuvent ou non être uniques pour les éléments sélectionnés, mais la combinaison de tous les champs doit être unique.

# Example

XML :

```
<root xmlns="namespace1">
  <A>
    <!-- if the ref-number is not equal to one of the key-number,
          the validation will give an error -->
    <part ref-number="1"/>
    <part ref-number="3"/>
  </A>
  <A>
    <!-- if the ref-number is not equal to one of the key-number,
          the validation will give an error -->
    <part ref-number="2"/>
  </A>
  <B>
    <part key-number="1"/>
    <part key-number="2"/>
    <part key-number="3"/>
  </B>
</root>
```

# Clé et référence de clé



```
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   targetNamespace="namespace1"
   xmlns:r="namespace1"
   elementFormDefault="qualified">

  <xs:element name="root">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="A" type="r:A"
                    maxOccurs="unbounded">
          <xs:keyref name="dummy"
                     refer="r:pNumKey">
            <xs:selector xpath="part"/>
            <xs:field xpath="@ref-number"/>
          </xs:keyref>
        </xs:element>
        <xs:element name="B" type="r:B"/>
      </xs:sequence>
    </xs:complexType>

    <xs:key name="pNumKey">
      <xs:selector xpath="r:B/r:part"/>
      <xs:field xpath="@key-number"/>
    </xs:key>
  </xs:element>
```

```
  <xs:complexType name="A">
    <xs:sequence>
      <xs:element name="part"
                  maxOccurs="unbounded">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="xs:string">
              <xs:attribute name="ref-number"
                            type="xs:integer"/>
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="B">
    <xs:sequence>
      <xs:element name="part"
                  maxOccurs="unbounded">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="xs:string">
              <xs:attribute name="key-number"
                            type="xs:integer"/>
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>

</xs:schema>
```

```
<!ELEMENT name (#PCDATA)>
```

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

```
<!ELEMENT name (#PCDATA)>
with attribute-list declarations
```

```
<xsd:element name="name">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attributeGroup ref="foo"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
```

```
<!ELEMENT name (#PCDATA | foo1 | foo2 | …)*>
with attribute-list declarations
```

```
<xsd:element name="name">
  <xsd:complexType mixed="true">
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:element ref="foo1"/>
      <xsd:element ref="foo2"/>
      …
    </xsd:choice>
    <xsd:attributeGroup ref="foo"/>
  </xsd:complexType>
</xsd:element>
```

```
<!ELEMENT name (element-content-model)>
with attribute-list declarations
```

```
<xsd:element name="name">
  <xsd:complexType>
    element-content-model
    <xsd:attributeGroup ref="foo"/>
  </xsd:complexType>
</xsd:element>
```

# Modèles de contenus

Référence des autres éléments dans le modèle de contenu

```
<xsd:element ref="name"/>
```

, dans le modèle de contenu

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

| dans le modèle de contenu

```
<xsd:choice>…</xsd:choice>
```

* dans le modèle de contenu

```
minOccurs="0" maxOccurs="unbounded"
```

+ dans le modèle de contenu

```
minOccurs="1" maxOccurs="unbounded"
```

? dans le modèle de contenu

```
minOccurs="0" maxOccurs="1"
```

# Déclarations des attributs

```
<!ATTLIST name
    attribute-declarations>
```

***attribute declaration***

#IMPLIED

#REQUIRED

CDATA

```
<xsd:attributeGroup name="name">
    attribute-declarations
</xsd:attributeGroup>
```

```
<xsd:attribute name="…" …>
    …
</xsd:attribute>
```

use="optional" ***(default)***

use="required"

type="xsd:string"

# Autres structures

```
<!-- … -->
```

```
<xsd:annotation>
  <xsd:documentation>
    …
  </xsd:documentation>
</xsd:annotation>
```

```
<!ENTITY % param-ent-name
    attribute-declarations>
```

```
<xsd:attributeGroup name="param-ent-name">
    attribute-declarations
</xsd:attributeGroup>
```

```
<!ENTITY % param-ent-name
    element-content-model>
```

```
<xsd:modelGroup name="param-ent-name">
    element-content-model
</xsd:modelGroup>
```

```
%param-ent-name;
```

```
<xsd:attributeGroup ref="param-ent-name"/>
```

```
%param-ent-name;
```

```
<xsd:modelGroup ref="param-ent-name"/>
```

```
<!ENTITY name …>
```

Déclarations d'entités parsées ou non

Non applicable