XML Schéma

Insuffisance des DTD

◆ Limitations

1. les DTD ne sont pas au format XML
 - Cela signifie qu'il est nécessaire d'utiliser un outil spécial pour "parser" un tel fichier, différent de celui utilisé pour l'édition du fichier XML
2. les DTD ne supportent pas les "espaces de nom"
 - En pratique, cela implique qu'il n'est pas possible, dans un fichier XML défini par une DTD, d'importer des définitions de balises définies par ailleurs
3. le "typage" des données est extrêmement limité
 - que du #PCDATA
 - d'où des efforts d'interprétation des données

XML Schéma

◆ Apports des schémas / DTD

- Un grand nombre de types de données de base
 - booléens, entiers, intervalles de temps, etc.
- Il est possible de créer de nouveaux types
 - par ajout de contraintes sur un type existant
 - des types de données utilisateurs qui vous permettent de créer votre propre type de données nommé
- La notion d'héritage
 - Les éléments peuvent hériter du contenu et des attributs d'un autre élément
- Une grande facilité de conception modulaire de schémas

XML Schéma

◆ Exemple : richesse apportée par le schéma

- Document XML : note.xml

```
<?xml version="1.0"?>
<note> <to>Tove</to>
      <from>Jani</from>
      <heading>Reminder</heading>
      <body>Don't forget me this weekend!</body>
</note>
```

- DTD pour ce document : note.dtd

```
<!ELEMENT note (to, from, heading, body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
  <!ELEMENT heading (#PCDATA)>
  <!ELEMENT body (#PCDATA)>
```

XML Schéma

- ◆ Un schéma pour ce document : apprécier la richesse :
note.xsd

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

```
<xs:schema ... >
```

```
  <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:complexType>
```

```
  </xs:element>
```

```
</xs:schema>
```

XML Schéma

Les premiers pas

◆ Structure de base

- Comme tout document XML, un Schema XML commence par un prologue et a un élément racine

```
<?xml version="1.0" ?>  
<xsd:schema ...>  
  <!-- déclarations d'éléments, d'attributs et de types ici -->  
  ...  
</xsd:schema>
```

- L'élément racine est l'élément **xsd:schema**
 - Cet élément racine est accompagné d'attributs qui précisent le lieu de définition des éléments
- Tout élément d'un schéma doit commencer par le préfixe **xsd** ou **XS**

XML Schéma

Les premiers pas

◆ Référencement de schema depuis un document XML

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

```
<note
```

```
  xmlns="http://www.w3schools.com"
```

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

```
  xsi:schemaLocation="http://www.w3schools.com note.xsd">
```

ou bien plus souvent utiliser :

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

```
xsi:noNamespaceSchemaLocation="note.xsd"
```

```
<to>Tove</to>
```

```
<from>Jani</from>
```

```
<heading>Reminder</heading>
```

```
<body>Don't forget me this weekend!</body>
```

```
</note>
```

XML Schéma

Les premiers pas

- `xmlns="http://www.w3schools.com"`
 - Indique au validateur du schéma l'endroit (espace) où sont définis les éléments
- `xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"`
 - Précise l'instance du schéma à utiliser
- `xsi:schemaLocation="http://www.w3schools.com note.xsd"`
 - Précise la localisation du fichier schéma par deux attributs
 - Le premier indique l'espace de nom
 - Le second indique le fichier contenant le schéma

XML Schéma

◆ Déclaration d'éléments

- Un élément, dans un schéma, se déclare avec la balise `<xsd:element>`
- Exemple :

```
<?xml version="1.0" ?>
<xsd:schema ...>
  <xsd:element name="remarque" type="xsd:string"/>
  <xsd:element name="contacts" type="typeContacts"/>
  <!-- déclarations de types ici -->
</xsd:schema>
```

◆ Ce schéma déclare deux éléments :

- `remarque` et `contacts`
- A chaque élément est associé un type via l'attribut `type`
 - `remarque` de type `xsd:string`, type simple prédéfini de XML Schema
 - `contacts` de type `typeContacts`, type complexe défini par l'utilisateur

XML Schéma

◆ Les types de données

- XML Schema permet donc de spécifier des types de données bien plus finement que le langage DTD
- Il distingue notamment deux types :
 - types simples
 - types complexes

◆ Types simples

- Ne peuvent comporter ni attributs, ni éléments enfants
- Il en existe de nombreux, prédéfinis, mais il est également possible d'en "dériver" de nouveaux
- Syntaxe de déclaration

```
<xs:element name="xxx" type="yyy"/>
```

XML Schéma

◆ Types simples : exemple

- Document XML

```
<lastname>Refsnes</lastname>
```

```
<age>34</age>
```

```
<dateborn>1968-03-27</dateborn>
```

- Schéma associé

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

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

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

Les types simples (1)

- ◆ string
 - Confirm this is electric
- ◆ normalizedString
 - Confirm this is electric
- ◆ token
 - Confirm this is electric
- ◆ byte
 - -1, 126
- ◆ unsignedByte
 - 0, 126
- ◆ base64Binary
 - GpM7
- ◆ hexBinary
 - 0FB7
- ◆ integer
 - -126789, -1, 0, 1, 126789
- ◆ positiveInteger
 - 1, 126789
- ◆ negativeInteger
 - -126789, -1
- ◆ nonNegativeInteger
 - 0, 1, 126789
- ◆ nonPositiveInteger
 - -126789, -1, 0
- ◆ int
 - -1, 126789675
- ◆ unsignedInt
 - 0, 1267896754

Les types simples (2)

- ◆ long
 - -1, 12678967543233
- ◆ unsignedLong
 - 0, 12678967543233
- ◆ short
 - -1, 12678
- ◆ unsignedShort
 - 0, 12678
- ◆ decimal
 - -1.23, 0, 123.4, 1000.00
- ◆ float
 - -INF, -1E4, -0, 0, 12.78E-2, 12, INF, NaN
- ◆ double
 - -INF, -1E4, -0, 0, 12.78E-2, 12, INF, NaN
- ◆ boolean
 - true, false 1, 0
- ◆ time
 - 13:20:00.000, 13:20:00.000-05:00
- ◆ dateTime
 - 1999-05-31T13:20:00.000-05:00
- ◆ duration
 - P1Y2M3DT10H30M12.3S
- ◆ date
 - 1999-05-31
- ◆ gMonth
 - --05--
- ◆ gYear
 - 1999

Les types simples (3)

- ◆ gYearMonth
 - 1999-02
- ◆ gDay
 - ---31
- ◆ gMonthDay
 - --05-31
- ◆ Name
 - shipTo
- ◆ QName
 - po:USAddress
- ◆ NCName
 - USAddress
- ◆ anyURI
 - <http://www.example.com/>,
 - <http://www.example.com/doc.html#ID5>
- ◆ language
 - en-GB, en-US, fr
- ◆ ID
 - "A212"
- ◆ IDREF
 - "A212"
- ◆ IDREFS
 - "A212" "B213"
- ◆ ENTITY
- ◆ ENTITIES
- ◆ NOTATION
- ◆ NMTOKEN, NMTOKENS
 - US
 - Brésil Canada Mexique

XML Schéma

Type simple

◆ Valeur par défaut et valeur fixée

- Un élément simple peut avoir une valeur par défaut ou une valeur fixée
 - Une valeur par défaut est assignée si aucune autre valeur n'est affectée
`<xs:element name="color" type="xs:string" default="red"/>`
 - Une valeur fixée est aussi assignée mais peut être changée
`<xs:element name="color" type="xs:string" fixed="red"/>`

XML Schéma

◆ Déclaration d'attributs

- Les attributs sont déclarés de types simples

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

- Seuls les éléments complexes peuvent avoir des attributs

- Exemple d'élément XML

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

- Voici la définition correspondante de type simple

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

- On peut également affecter une valeur par défaut

```
<xs:attribute name="lang" type="xs:string" default="EN"/>
```

- On peut également affecter une valeur fixe

```
<xs:attribute name="lang" type="xs:string" fixed="EN"/>
```

XML Schéma

◆ Déclaration d'attributs

- Tous les attributs sont optionnels par défaut, on peut cependant les expliciter :
 - `<xs:attribute name="lang" type="xs:string" use="optional"/>`
- On peut les rendre obligatoires
 - `<xs:attribute name="lang" type="xs:string" use="required"/>`

XML Schéma

◆ Déclaration d'attributs : restriction

- Des restrictions sont utilisées sur les attributs pour ne tolérer que des valeurs acceptables
- Les restrictions en XML sont appelées **facettes**

◆ Restriction à des valeurs

- Cet exemple définit un élément age avec une restriction : la valeur doit être comprise entre 0 et 100

```
<xs:element name="age">  
  <xs:simpleType>  
    <xs:restriction base="xs:integer">  
      <xs:minInclusive value="0"/>  
      <xs:maxInclusive value="100"/>  
    </xs:restriction>  
  </xs:simpleType>  
</xs:element>
```

XML Schéma

◆ Restriction à un ensemble de valeurs

- Pour limiter l'étendue d'un élément de XML à un ensemble de valeurs acceptables, on a l'habitude d'utiliser l'énumération
- Cet exemple montre l'élément "car" avec restriction à "Audi", "Golf" et "BMW"

```
<xs:element name="car">  
  <xs:simpleType>  
    <xs:restriction base="xs:string">  
      <xs:enumeration value="Audi"/>  
      <xs:enumeration value="Golf"/>  
      <xs:enumeration value="BMW"/>  
    </xs:restriction>  
  </xs:simpleType>  
</xs:element>
```

XML Schéma

◆ Restriction à un ensemble de valeurs

- On peut également l'écrire comme suit :

```
<xs:element name="car" type="carType"/>
<xs:simpleType name="carType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Audi"/>
    <xs:enumeration value="Golf"/>
    <xs:enumeration value="BMW"/>
  </xs:restriction>
</xs:simpleType>
```

- Dans ce cas-ci le type "carType" peut être employé par d'autres éléments parce que ce n'est pas une partie de l'élément "voiture"

XML Schéma

◆ Restriction à une série de valeurs

- Restriction de "letter" à des lettres minuscules :

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

XML Schéma

◆ Restriction à une série de valeurs

- Restriction de "initials" à trois lettres majuscules :

```
<xs:element name="initials">  
  <xs:simpleType>  
    <xs:restriction base="xs:string">  
      <xs:pattern value="[A-Z][A-Z][A-Z]"/>  
    </xs:restriction>  
  </xs:simpleType>  
</xs:element>
```

XML Schéma

◆ Restriction à une série de valeurs

- Restriction de "choice" à l'une des trois lettres x, y ou z :

```
<xs:element name="choice">  
  <xs:simpleType>  
    <xs:restriction base="xs:string">  
      <xs:pattern value="[xyz]"/>  
    </xs:restriction>  
  </xs:simpleType>  
</xs:element>
```

XML Schéma

Résumé des contraintes sur les types de données

Constraint	Description
enumeration	Defines a list of acceptable values
fractionDigits	Specifies the maximum number of decimal places allowed. Must be equal to or greater than zero
length	Specifies the exact number of characters or list items allowed. 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 (the value must be less than or equal to this value)
maxLength	Specifies the maximum number of characters or list items allowed. Must be equal to or greater than zero
minExclusive	Specifies the lower bounds for numeric values (the value must be greater than this value)
minInclusive	Specifies the lower bounds for numeric values (the value must be greater than or equal to this value)
minLength	Specifies the minimum number of characters or list items allowed. 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

XML Schéma

Composition de types simples

◆ Liste

- Les types listes sont des suites de types simples (ou *atomiques*).
- XML Schema possède trois types de listes intégrés : NMTOKENS (suite séparée par des blancs), ENTITIES et IDREFS
- Il est également possible de créer une liste personnalisée, par "dérivation" de types existants

- Par exemple :

```
<xsd:simpleType name="numéroDeTéléphone">  
  <xsd:list itemType="xsd:unsignedByte" />  
</xsd:simpleType>
```

- Un élément conforme à cette déclaration serait :

```
<téléphone>01 44 27 60 11</téléphone>
```

XML Schéma

◆ Union

- Les listes et les types simples intégrés ne permettent pas de choisir le type de contenu d'un élément
- On peut désirer, par exemple, qu'un type autorise soit un nombre, soit une chaîne de caractères particuliers
- Il est possible de le faire à l'aide d'une déclaration d'union
- Par exemple, sous réserve que le type simple numéroDeTéléphone ait été préalablement défini (voir précédemment), on peut déclarer...

```
<xsd:simpleType  
  name="numéroDeTéléphoneMnémonique">  
  <xsd:union memberTypes="xsd:string numéroDeTéléphone" />  
</xsd:simpleType>
```

- Les éléments suivants sont alors des "instances" valides de cette déclaration

```
<téléphone>18</téléphone>  
<téléphone>Pompiers</téléphone>
```

XML Schéma

◆ Types ou éléments complexes

- Un élément de type simple ne peut contenir de sous-éléments, et des attributs, d'où l'intérêt des éléments complexes
- Il y a quatre sortes d'éléments complexes :
 - Éléments vides
 - Éléments qui contiennent d'autres éléments
 - Éléments qui contiennent seulement du texte
 - Éléments qui contiennent d'autres éléments et du texte

XML Schéma

◆ Exemples d'éléments complexes

- Élément vide

```
<product pid="1345"/>
```

- Élément contenant d'autres éléments

```
<employee>
```

```
  <firstname>John</firstname>
```

```
  <lastname>Smith</lastname>
```

```
</employee>
```

- Élément contenant que du texte

```
<food type="dessert">Ice cream</food>
```

- Élément contenant des éléments et du texte

```
<description> It happened on
```

```
  <date lang="norwegian">03.03.99</date> ....
```

```
</description>
```

XML Schéma

◆ Comment définir un élément complexe ?

- Observons cet élément complexe qui contient d'autres éléments

```
<employee>
```

```
  <firstname>John</firstname>
```

```
  <lastname>Smith</lastname>
```

```
</employee>
```

- On peut définir l'élément dans un schéma de différentes manières

XML Schéma

1. L'élément "employee" peut être déclaré directement en distinguant ses composants, comme ceci:

```
<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>
```

- Mais ceci
 - Ne conduit que "employee" à utiliser l'appellation `complexType`
 - Les enfants "firstname" et "lastname" sont entourés de "sequence" les conduisant à être utilisés dans l'ordre où ils apparaissent

XML Schéma

2. L'élément "employee" peut avoir un type d'attribut qui se réfère au nom du type complexe à utiliser :

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

- En utilisant cette méthode
 - Plusieurs éléments peuvent ainsi se référer au type complexe "personinfo", comme ceci :

XML Schéma

Exemple :

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

XML Schéma

- On peut également se baser sur un élément d'un type complexe existant et lui ajouter des éléments :

```
<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> // définit des extensions ou des restrictions sur complexType
```

```
      <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>
```

"employee" doit contenir, dans l'ordre :
"firstname", "lastname", "adresse", "ville et
"pays"

XML Schéma

- ◆ Comment définir un type complexe pour un élément vide ?
 - Exemple
 - `<product prodid="1345" />`
 - L'élément "product" ci-dessus n'a aucun contenu
 - Pour définir un type sans contenu, nous devons définir un type qui permet seulement des éléments dans son contenu, mais nous ne déclarons réellement aucun élément, comme ceci :

XML Schéma

- Exemple

```
<xs:element name="product">  
  <xs:complexType>  
    <xs:complexContent>  
      <xs:restriction base="xs:integer">  
        <xs:attribute name="prodid" type="xs:positiveInteger"/>  
      </xs:restriction>  
    </xs:complexContent>  
  </xs:complexType>  
</xs:element>
```

XML Schéma

- On peut déclarer l'élément "product" de manière plus compacte :

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

XML Schéma

- On peut également donner à l'élément de complexType un nom, et laisser l'élément "product" avoir un attribut de type qui se rapporte au nom du complexType

→ si on emploie cette méthode, plusieurs éléments peuvent se rapporter au même type complexe :

```
<xs:element name="product" type="prodtype"/>
```

```
<xs:complexType name="prodtype">
```

```
<xs:attribute name="prodid" type="xs:positiveInteger"/>
```

```
</xs:complexType>
```

XML Schéma

◆ Type complexe à éléments

- Un type complexe à éléments est un type complexe contenant un élément qui contient seulement d'autres éléments
- Exemple : élément "person" contenant d'autres éléments

```
<person>  
  <firstname>John</firstname>  
  <lastname>Smith</lastname>  
</person>
```

- On peut définir l'élément "person" dans un schéma comme suit :

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

XML Schéma

Type complexe à éléments

- Sachant que la séquence conduit à utiliser les fils dans l'ordre où ils ont été déclarés, on peut proposer la solution suivante qui consiste à :
 - donner à l'élément de complexType un nom, et laisser l'élément "person" avoir un attribut de type qui se rapporte au nom du complexType
 - En employant cette méthode, plusieurs éléments peuvent se rapporter au même type complexe :

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

XML Schéma

◆ Type complexe à éléments textuels

- Ce type peut contenir des attributs et du texte, donc on ajoute un élément de simpleContent autour du contenu de manière à pouvoir faire des extensions ou des restrictions de ce contenu :
- Avec l'extension

```
<xs:element name="somename">  
  <xs:complexType>  
    <xs:simpleContent>  
      <xs:extension base="basetype">  
        .....  
      </xs:extension>  
    </xs:simpleContent>  
  </xs:complexType>  
</xs:element>
```

XML Schéma

- OU avec la restriction

```
<xs:element name="somename">
  <xs:complexType>
    <xs:simpleContent>
      <xs:restriction base="basetype">
        ....
      </xs:restriction>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

XML Schéma

- Nous pourrions également donner à l'élément de complexType un nom, et laisser "shoesize" avoir un attribut de type qui se rapporte au nom du complexType
 - si vous employez cette méthode, plusieurs éléments peuvent se rapporter au même type complexe :

```
<xs:element name="shoesize" type="shoetype"/>
  <xs:complexType name="shoetype">
    <xs:simpleContent>
      <xs:extension base="xs:integer">
        <xs:attribute name="country" type="xs:string" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
```

XML Schéma

◆ Type complexe à contenu mixte

- Il peut contenir des attributs, des éléments et du texte
- Exemple : l'élément "letter "

<letter>

Dear Mr.<name>John Smith</name>.

Your order <orderid>1032</orderid>

will be shipped on <shipdate>2001-07-13</shipdate>.

</letter>

- Notez que le texte qui paraît entre les éléments "name", "orderid " et "shipdate" sont tous les enfants de "letter"

XML Schéma

Type complexe à contenu mixte

- Le texte suivant déclare l'élément "letter" :

```
<xs:element name="letter">  
  <xs:complexType mixed="true" >  
    <xs:sequence>  
      <xs:element name="name" type="xs:string"/>  
      <xs:element name="orderid" type="xs:positiveInteger"/>  
      <xs:element name="shipdate" type="xs:date"/>  
    </xs:sequence>  
  </xs:complexType>  
</xs:element>
```

- Pour permettre à du texte d'apparaître entre les enfants-éléments de "letter", l'attribut **mixed** doit être à "vrai"
- L'étiquette `<xs:sequence>` signifie que les éléments définis (name, orderid et shipdate) doivent apparaître dans cet ordre à l'intérieur d'un élément "letter"

XML Schéma

Type complexe à contenu mixte

- Nous pourrions également donner à l'élément de complexType un nom, et laisser l'élément "letter" avoir un attribut de type qui se rapporte au nom du complexType
- Si on utilise cette méthode, plusieurs éléments peuvent se rapporter au même type complexe :

```
<xs:element name="letter" type="lettertype"/>
  <xs:complexType name="lettertype" mixed="true">
    <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="orderid" type="xs:positiveInteger"/>
      <xs:element name="shipdate" type="xs:date"/>
    </xs:sequence>
  </xs:complexType>
```

XML Schéma

◆ Type complexe avec indicateurs

- Les indicateurs permettent de contrôler l'usage des éléments à l'intérieur du type complexe
- Il existe 7 types d'indicateurs :
 - **Indicateurs d'ordre :**
 - All
 - Choice
 - Sequence
 - **Indicateurs d'occurrence :**
 - maxOccurs
 - minOccurs
 - **Indicateurs de groupes :**
 - Group name
 - attributeGroup name

XML Schéma

Type complexe avec indicateurs

◆ Indicateurs d'ordre

- Sont employés pour définir comment les éléments devraient se produire
- L'indicateur all
 - indique que les éléments enfant peuvent apparaître dans n'importe quel ordre et que chaque élément doit se produire une et une seule fois :

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

XML Schéma

Type complexe avec indicateurs

- L'indicateur choice
 - indique qu'un seul parmi les éléments enfants peut se produire:

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

XML Schéma

Type complexe avec indicateurs

- L'indicateur sequence
 - spécifie que les éléments enfants doivent apparaître dans un certain ordre :

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

XML Schéma

Type complexe avec indicateurs

◆ Indicateurs d'occurrence

- spécifie comment un élément peut se produire :
- L'indicateur maxOccurs :
 - indique le nombre maximum de fois qu'un élément peut se produire

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

- Le temps d'occurrence minimum par défaut est 1

XML Schéma

Type complexe avec indicateurs

- L'indicateur minOccurs :
 - indique le nombre minimum de fois qu'un élément peut se produire

```
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="full_name" type="xs:string"/>
      <xs:element name="child_name" type="xs:string"
        minOccurs="1" maxOccurs="10" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```
 - Pour permettre à un élément d'apparaître un nombre illimité de fois, utiliser un maxOccurs="unbounded"