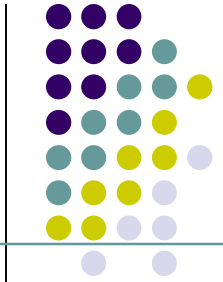


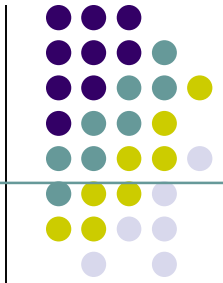
Université Gaston Berger de Saint-Louis
Année académique 2016-2017

XML Schema

References

1. XML Programming Success in a Day: Beginner's Guide to Fast, Easy, and Efficient Learning of XML Programming. Sam Key, 2015.
2. XML Programming: The Ultimate Guide to Fast, Easy, and Efficient Learning of XML Programming. Christopher Right, 2015
3. Beginning XML. Joe Fawcett and Danny Ayers, 2012.
4. World Wide Web School XML Tutorial: w3schools.com/xml
5. Research on youtube.com
6. Research on the internet.

Sommaire



- Introduction
- L'élément <schema>
- Les éléments simples
- Les attributs
- Restrictions
- Les éléments complexes
- Les indicateurs

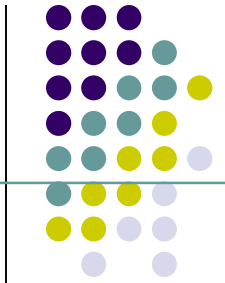
Introduction

Qu'est-ce qu'un schéma XML?

- Un schéma XML décrit la structure d'un document XML.
- C’est un langage appelé aussi XML Schema Definition (XSD).

Example

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="note">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="to" type="xs:string"/>
      <xs:element name="from" type="xs:string"/>
      <xs:element name="heading" type="xs:string"/>
      <xs:element name="body" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```



Introduction

Qu'est-ce qu'un schéma XML?

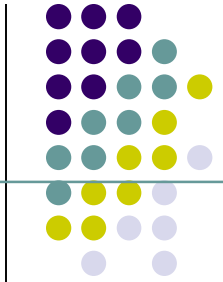
Il définit les blocs de construction légaux d'un document XML:

- les éléments et les attributs qui peuvent apparaître dans un document
- le nombre (et l'ordre) des éléments enfants
- Les types de données des éléments et des attributs
- les valeurs par défaut et fixes des éléments et des attributs

Pourquoi apprendre XML Schema?

- Dans le monde XML, des centaines de formats XML normalisés sont utilisés quotidiennement.
- Beaucoup de ces normes XML sont définis avec les schémas XML.
- XML Schema basé sur XML est plus puissant que son alternative DTD.

Introduction

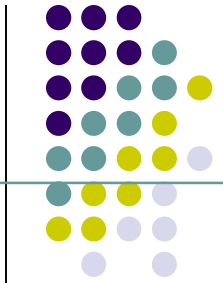


Types de données supportés

L'une des plus grande force de XML Schema est sa prise en charge des types de données.

- Il est plus facile de décrire le contenu du document admissible
- Il est plus facile de valider l'exactitude des données
- Il est facile de définir des restrictions sur les données
- Il est plus facile de définir des configurations de données (formats de données)
- Il est plus facile de convertir des données entre différents types de données

Introduction



Types de données supportés

Une autre grande force de XML Schema est qu'il est écrit en XML.

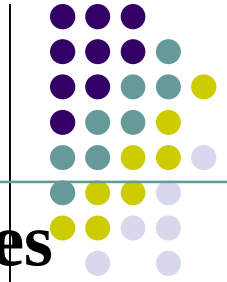
- On n’est pas obligé d'apprendre une nouveau langage
- On peut utiliser son éditeur XML pour modifier son schéma
- On peut utiliser son analyseur XML pour analyser son schéma
- On peut manipuler son schéma avec le DOM XML
- On peut transformer son schéma avec XSLT

XML Schema est extensible parce écrit en XML.

Avec une définition de schéma extensible, on peut:

- Réutiliser son schéma dans d'autres schémas
- Créer ses propres types de données dérivées des types standards
- Faire référence à plusieurs schémas dans le même document

Introduction



XML Schema sécurise la communication des données

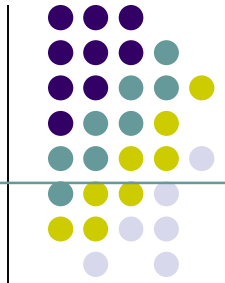
- Lors de l'envoi de données à partir d'un émetteur vers un récepteur, il est essentiel que les deux parties aient les mêmes «attentes» sur le contenu..
- Avec les schémas XML, l'expéditeur peut décrire les données d'une manière que le récepteur puisse comprendre.

  Exemple: Une date comme: "03-11-2000" sera, dans certains pays, interprété comme 3 novembre et dans d'autres pays comme 11 mars.
- Cependant, un élément XML avec un type de données comme ceci:

  <date type="date">2004-03-11</date>

  assure une compréhension mutuelle du contenu, parce que le type de données XML "date" exige le format "AAAA-MM-JJ".

Introduction



Bien formé ne suffit pas

Un document XML bien formé est un document qui est conforme aux règles de syntaxe XML, comme:

- il doit commencer par la déclaration XML
- il doit avoir un élément racine unique,
- Les balises de début doivent correspondre aux balises de fin
- Les éléments sont sensibles à la casse
- tous les éléments doivent être fermés
- tous les éléments doivent être correctement imbriquées
- toutes les valeurs d'attributs doivent être indiqués
- les entités doivent être utilisées pour les caractères spéciaux

Même si les documents sont bien formés, ils peuvent encore contenir des erreurs, et ces erreurs peuvent avoir des conséquences graves.

Avec les schémas XML, la plupart de ces erreurs peuvent être detecté par votre logiciel de validation.

Introduction

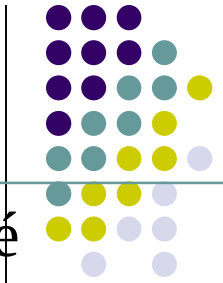
Utilisation

Un document XML peut avoir une référence à une DTD ou un schéma XML.

Exemple : Considérons le fichier XML nommé note.xml:

```
<?xml version="1.0"?>
<note>
  <to>Caam</to>
  <from>Yaaya</from>
  <heading>Reminder</heading>
  <body>Don't forget our meeting!</body>
</note>
```

Introduction



Exemple : L'exemple suivant est un fichier XML Schema appelé "note.xsd" qui définit les éléments du document XML ci-dessus ("note.xml")::

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

Introduction

- L'élément note est un **type complexe**, car il contient d'autres éléments.
- Les autres éléments (to, from, heading, body) sont des **types simples** car ils ne contiennent pas d'autres éléments..

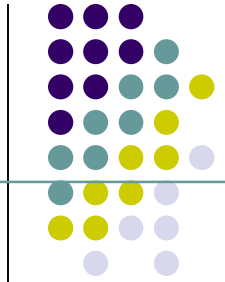
- **Une référence à un XML Schema**

Ce document XML à une référence à un schéma XML:

```
<?xml version="1.0"?>
<note xmlns="http://www.ugb.sn"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.ugb.sn note.xsd">

  <to>Caane</to>
  <from>Aysatu</from>
  <heading>Reminder</heading>
  <body>Don't forget the wedding!</body>
</note>
```

L'élément <schema>



Qu'est ce que l'élément <schema>?

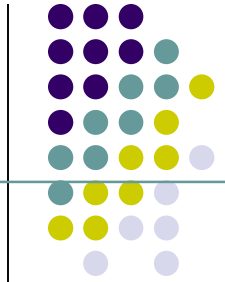
- L'élément <schéma> est l'élément racine de chaque schéma XML:

```
<?xml version="1.0"?>
<xs:schema>
...
</xs:schema>
```

- Il peut contenir certains attributs. Une déclaration de schéma ressemble souvent quelque chose comme ceci:

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

L'élément <schema>



Qu'est ce que l'élément <schema>?

- Le fragment de code suivant:

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

indique que les éléments et les types de données utilisés dans le schéma proviennent du namespace "http://www.w3.org/2001/XMLSchema".

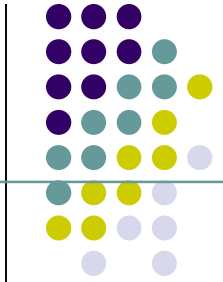
Il précise également que les éléments et les types de données venant de l'espace de noms "http://www.w3.org/2001/XMLSchema" devraient être préfixées avec **xs:**

- Le fragment de code suivant:

targetNamespace="http://www.ugb.sn"

indique que les éléments définis par ce schéma (note, to, from, heading, body.) proviennent de l'espace de noms "http://www.ugb.sn".

L’élément <schema>



- Le fragment de code :

xmlns="http://www.ugb.sn"

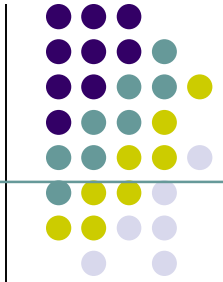
indique que l’espace de nom par défaut est "http://www.ugb.sn".

- Le fragment de code :

elementFormDefault="qualified"

indique que tous les éléments utilisés par le document XML qui sont déclarés dans ce schéma doivent être de namespace qualifié. C’est-à-dire que les éléments et les attributs sont dans le namespace cible (targetNamespace) du schéma

L'élément <schema>



- **Référencer un schéma dans un document XML**

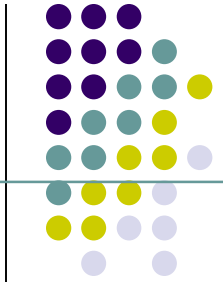
Ce document XML a une référence à un schéma XML:

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

<note xmlns="http://www.ugb.sn"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.ugb.sn note.xsd">

<to>Tuure</to>
<from>Jaañ</from>
<heading>Reminder</heading>
<body>Don't forget me our meeting!</body>
</note>
```

L'élément <schema>



- **Référencer un schéma dans un document XML**

Le fragment de code :

xmlns="http://www.ugb.sn"

spécifie la déclaration d'espace de noms par défaut. Cette déclaration indique le schéma-validateur que tous les éléments utilisés dans ce document XML sont déclarés dans le namespace "http://www.ugb.sn".

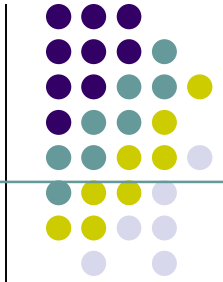
Une fois le namespace de l'instance du schéma XML est disponible:

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

On peut utiliser l'attribut schemaLocation. Cet attribute a deux valeurs, séparatées par un espace. La premiere est le namespace à utilizer et la seconde l'URL du schéma XML à utiliser pour ce namespace:
xsi:schemaLocation="http://www.ugb.sn note.xsd"

Les éléments simples

Qu’est ce qu’un Elément Simple?



- Un simple élément est un élément XML qui peut contenir seulement du texte. Il ne peut pas contenir d'autres éléments ou attributs.

- Cependant, la restriction "texte seulement" est tout à fait trompeur. Le texte peut être de plusieurs types différents. Il peut être l'un des types inclus dans la définition de schéma XML (boolean, string, date, etc.), ou un type personnalisé défini soi-même.

- On peut également ajouter des restrictions à un type de données afin de limiter son contenu, ou exiger que les données correspondent à un modèle spécifique.

Les éléments simples

Définition d'un élément simple

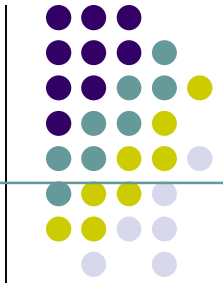
La syntaxe pour définir un élément simple est:

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

où xxx est le nom de l'élément et yyy est le type de la valeur de l'élément.

XML Schema a beaucoup de types de données intégrés dont les plus courants sont:

- xs:string
- xs:decimal
- xs:integer
- xs:boolean
- xs:date
- xs:time



Les éléments simples

Définition d'un élément simple

Exemple

Voici quelques éléments XML:

```
<lastname>Xaadim</lastname>
<age>5</age>
<dateborn>2011-11-27</dateborn>
```

Et les définitions correspondantes:

```
<xs:element name="lastname" type="xs:string"/>
<xs:element name="age" type="xs:integer"/>
<xs:element name="dateborn" type="xs:date"/>
```

Les éléments simples

Valeur par défaut et valeur fixe d'un élément simple

- Un élément simple peut avoir une valeur par défaut ou une valeur fixe spécifiée.

- Une valeur par défaut est automatiquement attribuée à l'élément quand aucune autre valeur n'est spécifiée.

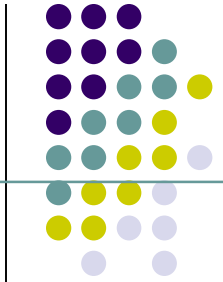
Exemple : Dans l'exemple suivant, la valeur par défaut est "red":

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

- Une valeur fixe est automatiquement assignée à l'élément et on ne peut pas spécifier une autre valeur..

Exemple : Dans cet exemple, la valeur fixe est "red":

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



Les attributs

Qu'est-ce qu'un attribut?

Un élément simple ne peut avoir d'attribut. Si un élément a des attributs, il est considéré comme étant de type complexe. Mais l'attribut lui-même est toujours déclaré comme un type simple.

Comment définir un attribut?

La syntaxe pour définir un attribut est:

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

où xxx est le nom de l'attribut et yyy le type de la valeur de l'attribut.

Exemple : Voici un élément XML avec un attribut:

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

Et voici la définition de l'attribut correspondant:

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

Les attributs

Valeur par défaut et valeur fixe des attributs

- Un attribut peut avoir une valeur par défaut ou une valeur fixe spécifiée.
- Une valeur par défaut est automatiquement affectée à l'attribut quand aucune autre valeur n’est spécifiée.

Dans cet exemple, la valeur par défaut est "EN":

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

- Une valeur fixe est automatiquement affectée à l'attribut, et on ne peut pas spécifier une autre valeur.

Dans cet exemple, la valeur fixe est "EN":

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

Les attributs

Attribut facultatif et attribut obligatoire

Un attribut est facultatif par défaut. Pour spécifier qu'il est requis, on utilise attribut "use":

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

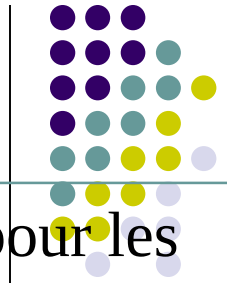
Restrictions sur les contenus

- Quand un élément XML ou attribut a un type de données défini, il met des restrictions sur le contenu de l'élément ou attribut.

- Si un élément XML est de type "xs: date" et contient une chaîne comme "Bonjour tout le monde", l'élément ne sera pas valide.

- Avec les schémas XML, on peut également ajouter ses propres restrictions à ses éléments et attributs XML. Ces restrictions sont appelées facettes. On en saura plus sur les facettes dans la suite.

Les restrictions

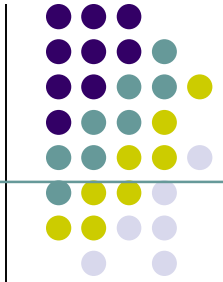


Une restriction est utilisée pour définir des valeurs acceptables pour les éléments ou les attributs XML. Les restrictions sur les éléments XML sont appelées facettes.

Restrictions sur les valeurs

L'exemple suivant définit un élément appelé «âge» avec une restriction. La valeur de l'âge ne peut pas être inférieure à 0 ou supérieure à 120:

```
<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>
```



Les restrictions

Restriction sur un ensemble de valeurs

Pour limiter le contenu d'un élément XML à un ensemble de valeurs, on doit utiliser la contrainte d'énumération.

L'exemple ci-dessous définit un élément appelé "voiture" avec une restriction. Les seules valeurs acceptables sont: Audi, Golf, 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>
```

Les restrictions

Restrictions sur un ensemble de valeurs

L'exemple ci-dessus pourrait également être écrit 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>
```

Remarque: Dans ce cas, le type "carType" peut être utilisé par d'autres éléments, car il ne fait pas partie de l'élément "voiture".

Les restrictions

Restrictions sur une série de valeurs

- Afin de limiter le contenu d'un élément XML à une série de chiffres ou de lettres qui peuvent être utilisés, on utilise la contrainte pattern.

- L'exemple ci-dessous définit un élément appelé "lettre" avec une restriction. La seule valeur acceptable est l'une des lettres en MINUSCULES de a à z:

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

Les restrictions

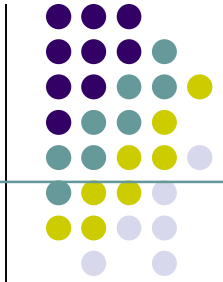
Restrictions sur une série de valeurs

L'exemple suivant définit un élément appelé "initials" avec une restriction. La seule valeur acceptable est trois des lettres MAJUSCULES de A à Z:

```
<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>
```

Les restrictions

Restrictions sur une série de valeurs



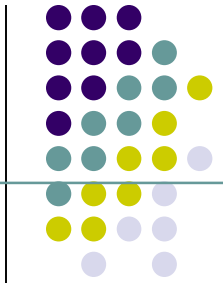
- L'exemple suivant définit également un élément appelé "initials" avec une restriction.

- La seule valeur acceptable est trois lettres majuscules ou minuscules de a à z:

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

Les restrictions

Restrictions sur une série de valeurs



- L'exemple suivant définit un élément appelé «choix» avec une restriction.

- La seule valeur acceptable est l'une des lettres suivantes: 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>
```

Les restrictions

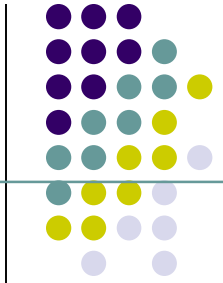
Restrictions sur une série de valeurs

- L'exemple suivant définit un élément appelé "prodid" avec une restriction.

- Les seules valeurs possibles sont les nombres de CINQ chiffres compris entre 0 à 9:

```
<xs:element name="prodid">
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:pattern value="[0-9][0-9][0-9][0-9][0-9]"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Elément complexe



Qu'est-ce qu'un élément complexe?

C’est un élément XML qui contient d'autres éléments et/ou des attributs.
Il existe quatre types d'éléments complexes:

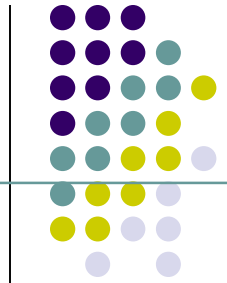
- les éléments vides
- les éléments ne contenant que d'autres éléments
- les éléments ne contenant que du texte
- les éléments contenant à la fois des éléments et du texte

Note: Chacun de ces éléments peut contenir des attributs aussi bien!

Exemple :
Un élément complexe, "produit", qui est vide:

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

Elément complexe



Exemple :

- Un élément complexe "employee" ne contenant que d'autres éléments:

```
<employee>
  <firstname>Jean</firstname>
  <lastname>Sarr</lastname>
</employee>
```

- Un élément complexe "food" ne contenant que du texte:

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

- Un élément complexe "description" contenant à la fois des éléments et du texte:

```
<description>
It happened on <date lang="wolof">03/03/99</date> ....
</description>
```

Elément complexe

Comment définir un élément complexe

Voici un élément complexe qui ne contient que d'autres éléments:

```
<employee>
  <firstname>Suun</firstname>
  <lastname>Fay</lastname>
</employee>
```

Un élément complexe est défini dans un schéma XML de deux façons différentes:

1. En déclarant directement le nom de l'élément, comme suit:

```
<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>
```

Elément complexe

Comment définir un élément complexe

- Si on utilise la méthode décrite ci-dessus, seul l'élément "employe" peut utiliser le type complexe spécifié.
- Notons que les éléments enfants, "firstname" and "lastname", sont entourés par l'indicateur <sequence>.
- Les éléments enfants doivent apparaître dans le même ordre qu'ils sont déclarés.

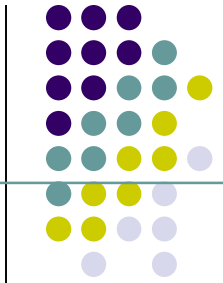
2. L'élément peut avoir un attribut type qui désigne le nom du type complexe à utiliser:

```
<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>
```

Elément complexe

Comment définir un élément complexe



Si on utilise la méthode décrite ci-dessus, plusieurs éléments peuvent se référer au même type complexe, comme ceci:

```
<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>
```

Elément complexe

Comment définir un élément complexe

Définir un élt complexe à partir d'un elt existant et en ajouter d'autres, comme suit

```
<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>
```

Elément complexe

Élément complexe vide

Un élément XML vide: <product prodid="1345" />

L'élément "product" ci-dessus n'a pas de contenu. Pour définir un type sans contenu, nous devons définir un type qui permet des éléments dans son contenu, mais sans y déclarons d'éléments, comme suit:

```
<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>
```

Elément complexe

Élément complexe vide

- Dans l'exemple ci-dessus, on définit un type complexe avec un contenu complexe.
- L'élément complexContent signale l'intention de restreindre ou d'étendre le modèle de contenu d'un type complexe, et la restriction au nombre entier déclare un attribut, mais ne présente aucun contenu d'élément.
- Cependant, il est possible de déclarer l'élément "product" plus compacte, comme suit :

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

Elément complexe

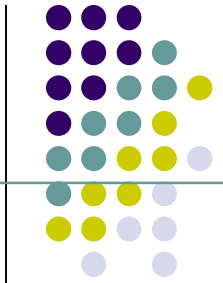
Élément complexe vide

On peut aussi donner à l'élément complexType un nom, et laisser l'élément "product" avoir un attribute type qui se réfère au nom du complexType (avec cette méthode, plusieurs éléments peuvent se référer à un même type complexe):

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

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

Elément complexe



Un type complexe contenant que des éléments

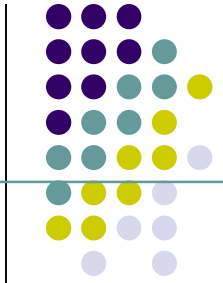
Avec l'élément, "person", qui ne contient que des éléments:

```
<person>
  <firstname>Charles</firstname>
  <lastname>Fay</lastname>
</person>
```

On peut définir l'élément "person" dans un schema 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>
```

Elément complexe



Un type complexe contenant que des éléments

- Notons que la balise <xs: sequence> spéficie que les éléments "prenom" et "nom" doivent apparaître dans cet ordre dans un élément "personne".

- On peut donner à l'élément complexType un nom, et laisser l'élément "personne" avoir un attribut type qui fait référence au nom du complexType (ainsi plusieurs éléments peuvent se référer 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>
```

Elément complexe

Élément complexe contenant que du texte

Ce type d'élément ne contient que du contenu simple (texte et attribut), donc on ajoute un élément simpleContent autour du contenu. Lors de l'utilisation de contenu simple, on doit définir une extension ou une restriction dans l'élément simpleContent, comme suit :

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

OR

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

Elément complexe

Élément complexe contenant que du texte

Conseil: On utilise l'élément d'extension / de restriction pour étendre ou restreindre le type simple de base de l'élément.

Voici un exemple d'élément XML, "shoesize", qui contient du texte uniquement:: <shoesize country="Gambia">35</shoesize>

L'exemple suivant déclare un complexType, "shoesize". :

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

Elément complexe

Élément complexe contenant que du texte

On pourrait aussi donner à l'élément complexType un nom, et laisser l'élément "shoesize" avoir un attribut type qui fait référence au nom de l'élément complexType (ainsi plusieurs éléments peuvent se référer 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>
```

Elément complexe

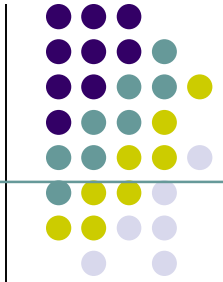
Types complexes avec un contenu mixte

Un élément de type complexe mixte peut contenir des attributs, des éléments et du texte.

L'élément "letter" suivant contient à la fois du texte et d'autres éléments:

```
<letter lang="en">
  Dear Mr.<name>Pierre Gomez</name>.
  Your order <orderid>1032</orderid>
  will be shipped on <shipdate>2001-07-13</shipdate>.
</letter>
```

Elément complexe



Types complexes avec un contenu mixte

Le schéma 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:attribute name=“lang" type="xs:string"/>
  </xs:complexType>
</xs:element>
```

Remarque: Pour permettre aux données caractères d’apparaître entre les éléments enfants de «letter», l'attribut mixed doit être à "true".

Elément complexe

Types complexes avec un contenu mixte

On pourrait aussi donner à l'élément complexType un nom, et laisser à l'élément "letter" avoir un attribut type qui fait référence au complexType (ainsi plusieurs éléments peuvent se référer 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:attribute name="lang" type="xs:string"/>
</xs:complexType>
```

Les indicateurs XSD

On peut contrôler la façon dont les éléments doivent être utilisés dans les documents avec des indicateurs. Il y a sept indicateurs:

Les indicateurs d'ordre:

- All
- Choice
- Sequence

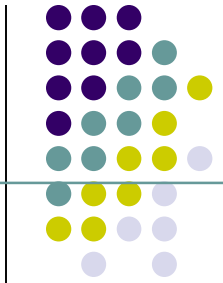
Les indicateurs d'occurrence :

- maxOccurs
- minOccurs

Les indicateurs de groupe :

- Group name
- attributeGroup name

Les indicateurs XSD



Les indicators d'ordre

Les indicateurs d'ordre sont utilisés pour définir l'ordre des éléments.

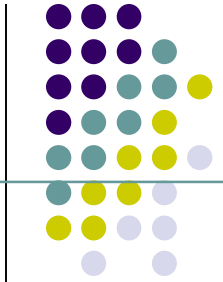
- **L'indicateur All**

Il specifie que les éléments enfants peuvent apparaître dans n'importe quel ordre mais avec une seule occurrence à la 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>
```

Remarque: Lorsqu'on utilise l' indicateur <all> on peut régler l'indicateur <minOccurs> à 0 ou 1 et l'indicateur <maxOccurs> ne peut être défini qu'à 1 (Les indicateurs <minOccurs> et <maxOccurs> sont décrits plus loin).

Les indicateurs XSD



Les indicators d'ordre

- **Indicateur Choice**

Il spécifie que c'est soit l'un soit l'autre des éléments enfant qui est choisi:

```
<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>
```

Les indicateurs XSD

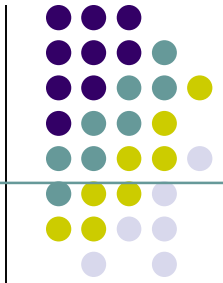
Les indicators d'ordre

- **L'indicateur Sequence**

Il spécifie que les éléments enfant doivent apparaître dans un ordre spécifique:

```
<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>
```

Les indicateurs XSD



Les indicators d'occurence

Ils sont utilisés pour définir la fréquence d'apparition d'un élément.

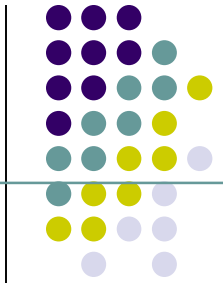
Remarque: Pour les indicateurs d'ordre et de groupe la valeur par défaut pour maxOccurs et minOccurs est 1.

- **L'indicator maxOccurs**

L'indicateur <maxOccurs> indique le nombre maximum de fois qu'un élément apparait:

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

Les indicateurs XSD



Les indicators d'occurence

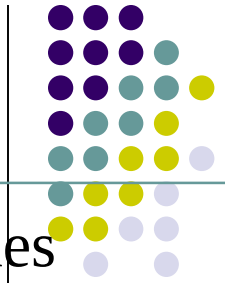
- **L'ndicateur minOccurs**

Il indique le nombre minimum de fois qu'un élément peut apparaître:

```
<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" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Remarque: Pour indiquer qu'un élément apparaisse un nombre illimité de fois, on utilise maxOccurs = "unbounded":

Les indicateurs XSD



Les indicateurs de groupe : ils sont utilisés pour définir des ensembles d'éléments associés.

- **Groupe d'élements**

Des éléments sont regroupés avec la declaration group comme suit:

```
<xs:group name="groupname">
...
</xs:group>
```

On peut définir un élément all, choice ou sequence dans la déclaration du groupe.

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

Les indicateurs XSD

Les indicateurs de groupe :

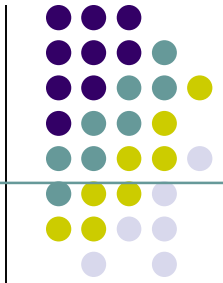
Groupe d'élements: une fois qu'on définit un groupe, on peut en faire référence dans une autre définition, comme suit:

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

<xs:element name="person" type="personinfo"/>

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

Les indicateurs XSD



Les indicateurs de groupe :

- **Groupe d'attributs**

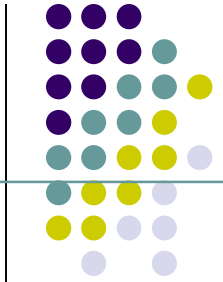
Un groupe d'attributs est définis par la déclaration de attributeGroup, comme suit :

```
<xs:attributeGroup name="groupname">
...
</xs:attributeGroup>
```

L'exemple suivant définit un groupe d'attributs appelé "personattrgroup":

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

Les indicateurs XSD



Les indicateurs de groupe :

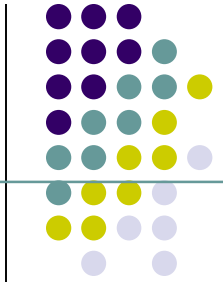
- **Groupe d'attributs**

Après avoir défini un groupe d'attributs, on peut le référencer dans une autre définition, comme suit:

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

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

Extension de XML



L’élément <any>

L'élément <any> permet d'étendre le document XML avec des éléments non spécifiés par le schéma.

Exemple: Considérons le fichier XSD "family.xsd" qui montre une déclaration de l'élément "personne". En utilisant l'élément <any> on peut étendre (après <lastname>) le contenu de "person" avec tout élément:

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

Extension de XML

L'élément <any>

Si on veut étendre l'élément "person" avec un élément enfant, on peut le faire même si l'auteur du schema ci-dessus n'a jamais declaré d'élément enfant. On utilise le fichier XSD "children.xsd" suivant:

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

<xs:element name="children">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="childname" type="xs:string“   maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

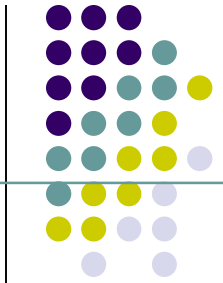
Extension de XML

L'élément <any>

Le fichier XML ci-dessous (appelé "Myfamily.xml"), utilise des composants provenant de deux schémas differents: "family.xsd" et "children.xsd":

```
<?xml version="1.0" encoding="UTF-8"?>
<persons xmlns="http://www.utg.gm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" nxsi:schemaLocation
="http://www.ugb.sn.com family.xsd http://www.utg.gm children.xsd">
<person>
  <firstname>Abdoulaye</firstname>
  <lastname>Wade</lastname>
  <children>
    <childname>Karim</childname>
  </children>
</person>
<person>
  <firstname>Abdou</firstname>
  <lastname>Diouf</lastname>
</person>
</persons>
```

Extension de XML



L' élément <anyAttribute>

Il permet d'étendre un document XML avec des attributs non specifies par la schéma.

L'exemple suivant est un fragment d'un schema XML appelé "family.xsd" qui présente une declaration d'un élément "person". En utilisant l'élément <anyAttribute> on peut ajouter tout nombre d'attribute à l'élément "person":

```
<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:anyAttribute/>
  </xs:complexType>
</xs:element>
```

Extension de XML

L' élément <anyAttribute>

Si on veut etendre l'élément "person" avec un attribut "gender", on pourrait le faire même si l'auteur du schema ci-dessus n'a jamais déclaré un attribut "gender". Considerons le fichier XSD suivant appelé "attribute.xsd":

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

<xs:attribute name="gender">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="male|female"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>

</xs:schema>
```

Extension de XML

L' élément <anyAttribute>

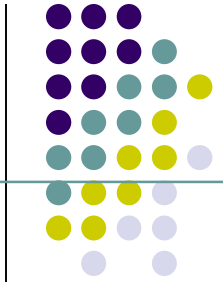
Le fichier XML suivant appelé "Myfamily.xml", utilize des composants provenants de deux schémas differents: "family.xsd" et "attribute.xsd":

```
<?xml version="1.0" encoding="UTF-8"?>
<persons xmlns="http://www.utg.gm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:SchemaLocation="http://www.utg.gm family.xsd
http://www.ugb.sn attribute.xsd">

<person gender="female">
  <firstname>Yaasin</firstname>
  <lastname>Mbóoj</lastname>
</person>

<person gender="male">
  <firstname>Góor</firstname>
  <lastname>Seen</lastname>
</person>
</persons>
```

Le fichier XML ci-contre est valide car le schema "family.xsd" permet d'ajouter un attribute à l'élément "person".
Les éléments <any> et <anyAttribute> sont utilisés pour créer des documents EXTENSIBLE! Ils permet aux documents de contenir des éléments additionnels qui n'ont pas été declarés dans le schema XML principal.



Merci pour votre attention