Système d'Information

< technologies XML >

[02] XML Schema – Partie I

Joël Dumoulin
joel.dumoulin@hefr.ch
+41 26 429 69 60

Jacky Casas, Leonardo Angelini, Omar Abou Khaled
jacky.casas@hefr.ch, leonardo.angelini@hefr.ch, omar.aboukhaled@hefr.ch

HumanTech
Technology for
Human Wellbeing Institute

Buts et plan

- Rappel des principes généraux des DTD
  - Rôle et contexte d'utilisation
  - Les limites des DTD
- Introduction aux XML Schémas
  - Présentation
  - Avantages et inconvénients
- Définition d'un XML Schéma
  - Attachement à un document XML
  - Composants
    - ✓ Déclaration des éléments et des attributs
    - ✓ Définition des types simples et complexes
  - Structure d'un schéma XML
    - ✓ Déclaration locale
    - ✓ Déclaration globale

DTD (rappel)

- Une DTD définit un modèle de document

- Les fonctions de base d'une DTD
  - Définir la **structuration** des documents XML
  - Définir le modèle de contenu des **éléments**
  - Déclarer une liste **d'attributs** pour un élément

- Plusieurs types de déclarations
  - des éléments, des attributs, des entités, et des notations

DTD vs XML Schéma

- Problème au niveau de **typage** de données:
  - Comment garantir que <Price> contient un nombre?
- Solution
  - DTD
    - ✓ Construire la logique de validation dans l'application
  - XML Schema
    - ✓ Déclarer le type de données dans le schéma
    - ✓ Validation avec un module générique

```
<Book ISBN="1-558-28592-X">
  <Title>XML: A Primer</Title>
  <Price>24.99</Price>
  <Author>
    <Name>Simon St. Laurent</Name>
    <ContactInfo>
      <Email>sst@laurent.com</Email>
    </ContactInfo>
  </Author>
</Book>
```

Les limites avec les DTD

- Inconvénients des DTD
  - Les DTD offrent des possibilités de typage des données très limitées
  - Les contraintes de répétitions sont limitées
    - ✓ Ex: spécifier qu'un élément ne peut apparaître que deux fois
    - ✓ Les éléments dans une DTDs sont limités à 0, 1 ou infinité d'occurrences
  - Les DTD ne sont pas définies comme des documents XML
    - ✓ Deux syntaxes différentes
  - Les DTD ne sont pas extensibles

Qu'est ce que XML Schéma ? 1/2

- Définition d'un schéma:
  - Dans le cadre des bases de données
    - ✓ Terme se référant à l'organisation et la structure d'une base de données
  - Dans le cadre de XML
    - ✓ Une définition formelle pour une classe de documents

- Un schéma XML fournit une description de documents XML en utilisant également XML !

- XML Schéma est un standard défini par le consortium W3C
  - http://www.w3.org/XML/Schema

Qu'est ce que XML Schéma ? 2/2

- Les schémas XML sont analogues aux DTD.

  Cependant, ils sont:

  - Plus performants
    - ✓ Typage des données
    - ✓ Définition précise des occurrences
  - Plus flexibles
    - ✓ Possibilités d'extensions et de réutilisations
  - Plus complexes

Objectifs généraux d'un schéma

- Définir la structure et les contenus des documents
- Fournir un ensemble de types primitifs pour les données
- Permettre l'évolution des schémas
- Définir des descriptions et des contraintes spécifiques
- Un document XML doit pouvoir être validé relativement à son schéma

Les composants d'un schéma XML

- Types de composants:
  - **Déclaration** d'éléments ou d'attributs
  - **Définition** de types simples ou complexes

  - Définitions de modèles de groupes d'éléments
  - Définition de groupe d'attributs
  - Définition de contraintes d'identifiants

  - Déclaration de notations (équivalent à celle de la DTD)
  - Annotations, Wild-cards,…

Déclarations vs Définitions

- La déclaration définit un élément ou un attribut en spécifiant le nom et les types de données pour ce composant.

- Une définition décrit un type complexe ou simple contenant éléments ou attributs pouvant être définis ailleurs dans le document.

Un premier exemple ...

email.xml

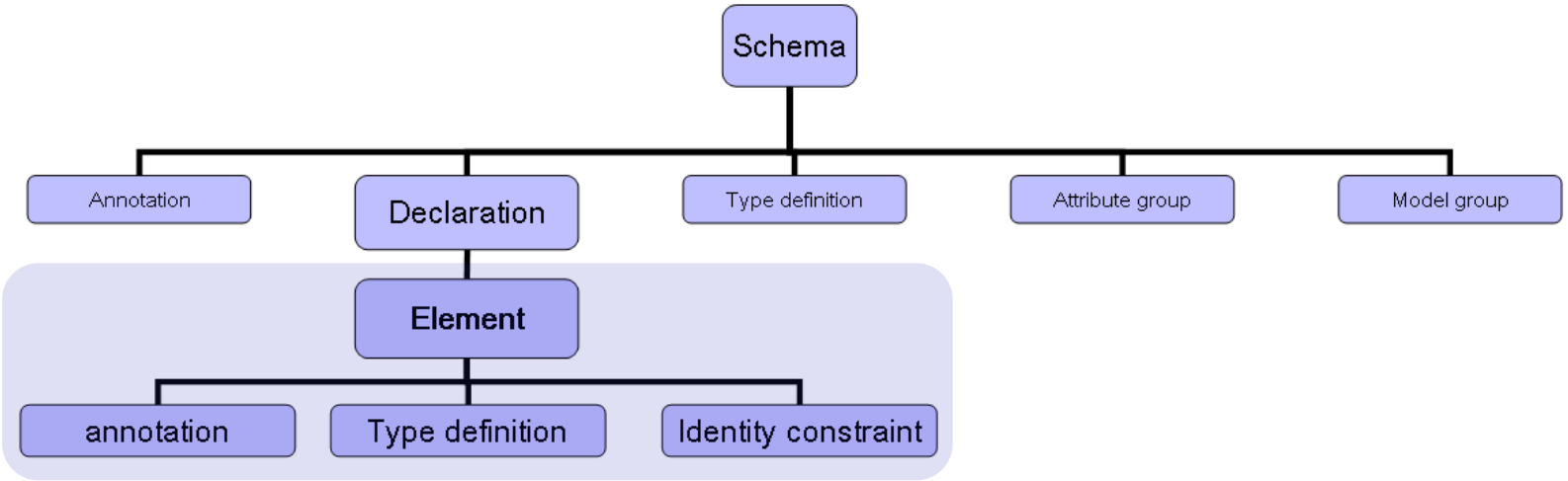
```
<?xml version = "1.0"?>
<incomingEmail xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:noNamespaceSchemaLocation="email.xsd">
  <from>john.milton@company.com</from>
  <subject>XML Schema workshop</subject>
  <message>Enjoy the workshop!</message>
</incomingEmail>
```

email.xsd

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

Le déclaration d'éléments 1/2

- Une déclaration d'élément peut contenir:
  - Une *annotation* fournissant une information pour les êtres humain
  - Une *identity constraint* pour définir l'unicité ou des références.
  - Un *type definition* (simple ou complexe) pour définir le contenu de l'élément (attributs, éléments, données…)

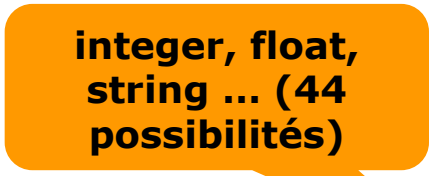


La déclaration d'éléments 2/2

XML DTD:

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

XML Schema:




```
<xs:element name="element-name" type="simple-type" />
```

- **Name = Nom de l'élément**
- **Type = Type de données de l'élément**

Element Occurrence Indicators

DTD Cardinality Operator	minOccurs Value	maxOccurs Value	Number of Child Element(s)
[none]			
?			
*			
+			

```
<xs:element name="to" type="xs:string" minOccurs="1" maxOccurs="1"/> <!-- default -->

<xs:element name="to" type="xs:string"/>                             <!-- default -->

<xs:element name="to" type="xs:string" maxOccurs="3"/>

<xs:element name="to" type="xs:string" minOccurs="3"/>

<xs:element name="to" type="xs:string" minOccurs="1" maxOccurs="5"/>

<xs:element name="to" type="xs:string" minOccurs="2" maxOccurs="unbounded"/>
```

La déclaration d'attributs

XML DTD:

```
<!ATTLIST element-name attribute-name ...>
```

XML Schema:

integer, float, string ...

```
<xs:attribute name="attribute-name" type="simple-type" />
```

- Name = Nom de l'attribut
- Type = Type de données de l'attribut

La déclaration d'attributs (exemple)

```
<?xml version = "1.0"?>
<incomingEmail opened="false">
  <from>john.milton@company.com</from>
  <subject>XML Schema workshop</subject>
  <message>Enjoy the workshop!</message>
</incomingEmail>
```

```
<?xml version = "1.0"?>
<xs:schema xmlns:xs = "http://www.w3.org/2001/XMLSchema">
  <xs:element name = "incomingEmail">
    <xs:complexType>
      <xs:sequence>
        <xs:element name = "from" type = "xs:string"/>
        <xs:element name = "subject" type = "xs:string"/>
        <xs:element name = "message" type = "xs:string"/>
      </xs:sequence>
      <xs:attribute name = "opened" type = "xs:boolean"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Occurrences et valeurs par défaut

- L'attribut **use** peut prendre les valeurs suivantes dans des déclarations d'attributs:

```
<xs:attribute name="opened" type="xs:boolean" use="required"/>
```

```
<xs:attribute name="opened" type="xs:boolean" use="optional"/>
```

```
<xs:attribute name="opened" type="xs:boolean" use="prohibited"/>
```

- L'attribut **default** permet de définir la valeur par défaut de l'attribut:

```
<xs:attribute name="opened" type="xs:boolean" default="true"/>
```

Les possibilités de déclarations

- Trois moyens:
  - Inline Declarations
    - ✓ Elle est définie **localement**, au moment où on a besoin d'un élément ou d'un attribut, à l'intérieur d'un autre composant du schéma
  - Referencing Declaration
    - ✓ Définition de la déclaration d'une manières **globale** et ensuite référencer cette déclaration depuis une autre définition
  - Named Types
    - ✓ C'est la méthode de nommage du type de définition pour qu'il soit référencé ensuite dans le document.
    - ✓ Déclaration une fois et utilisation plusieurs fois dans le document (Semblable au principe des "entities" avec la DTD)

Types de déclarations (exemples I)

Inline declaration

L'élément « firstName » est déclaré directement dans la définition de l'élément « employees ».

L'élément « firstName » ne peut pas être réutilisé ailleurs dans le document.

```
<xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="employees">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="firstName" type="xs:string"/>
        <xs:element ref="lastName"/>
        <xs:element name="address" type="addressType"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

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

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

</xs:schema>
```

Types de déclarations (exemples II)

Referencing declaration

L'élément « lastName » est déclaré de manière globale. Il est ensuite référencé depuis un autre endroit dans le document.

L'élément « lastName » peut être réutilisé ailleurs dans le document ou dans d'autres documents.

```
<xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="employees">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="firstName" type="xs:string"/>
        <xs:element ref="lastName"/>
        <xs:element name="address" type="addressType"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

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

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

</xs:schema>
```

Types de déclarations (exemples III)

Named types

Les éléments formant une adresse sont regroupés dans un type nommé « addressType ».

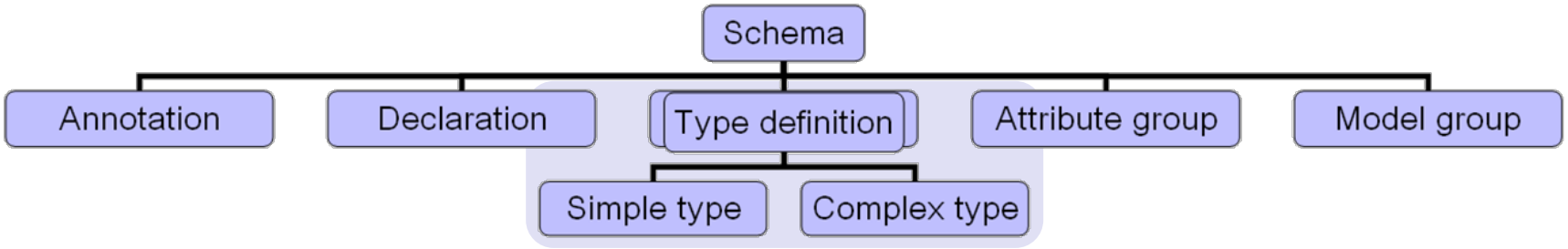
L'élément « address » référence ensuite ce type à l'aide de son attribut « type ».

Le type « addressType » peut être réutilisé ou modifié ailleurs dans le document ou dans d'autres documents.

```
<xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="employees">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="firstName" type="xs:string"/>
        <xs:element ref="lastName"/>
        <xs:element name="address" type="addressType"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

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

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

</xs:schema>
```

Les définitions de types

- Il existe deux types de définitions et chaque type peut être définit de deux manières:
  - **Simple type definition** est utilisé pour définir les attributs et les éléments contenant "uniquement" des données
  - **Complex type definition** définit le modèle de contenu pour les éléments contenant des éléments et d'attributs



Les définitions de types

- On peut les définir
  - **Comme "root level"** et dans ce cas il faut les nommer et les référencer ultérieurement dans le document ("named types")
  - Comme **imbriqué avec d'autres éléments** ("anonymous types") et dans ce cas il ne faut pas les référencer

Définition locale vs globale

Anonymous types

Le type est défini de manière locale (imbriqué dans la déclaration de l'élément). De ce fait, il ne peut être utilisé que pour l'élément dans lequel il a été défini.

Named types

Le type est défini de manière global (au niveau « root »). De ce fait, il peut être référencé depuis plusieurs autres endroits dans le document.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="employees">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="firstName" type="xs:string"/>
        <xs:element ref="lastName"/>
        <xs:element name="address" type="addressType"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

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

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

</xs:schema>
```

La définition de types simples

- Il existe **44 types** de données qui peuvent être utilisées avec les schémas XML

- XML schéma offre la possibilité de définir ses propres types simples.
  - Les nouveaux types sont basés sur les types simples existants
  - Possibilités de restreindre les valeurs possibles suivant différents aspects (facettes)
    - ✓ Longueur maximale
    - ✓ Valeurs (minimale, maximale, énumération)
    - ✓ Pattern (format)
  - Définition de listes de valeurs
  - Union de plusieurs types

Exemples de types simples 1/2

Primitive type	Description	Example
`string`	represents any legal character strings that matches the Char production in XML 1.0 Recommendation	`Software AG`

`anyURI`	represents a URI	`http://www.softwareag.com`
		`mailto://info@softwareag.com`
		`mySchema.xsd`

`boolean`	represents binary logic, true or false	`true`, `false`, `1`, `0`
`decimal`	represents arbitrary precision decimal numbers	`3.141` + and – is used to represent positive and negative numbers
`float`	real number corresponding to a single precision 32 bit floating point type	`-INF`, `-1E4`, `4.5E-2`, `37`, `INF`, `NAN`

Exemples de types simples 2/2

Primitive type	Description	Example
duration	represents a duration of time in the format `PnYnMnDTnHnMnS`	`P1Y0M1DT10H15M10S`
time	represents an instance of time that occurs every day in the format `HH:MM:SS`	`16:43:20`
date	represents a calendar date from the Gregorian calendar in the format `CCYY-MM-DD`	`2001-09-03`
integer	standard mathematical concept of `integer` numbers	`-5, 0, 9, 23`
positiveInteger	An `integer` of 1 or higher	`1, 43, 7532`

Ex:

```
<xs:element name=”publicationDate" type="xs:date"/>
<xs:element name=”author" type="xs:string"/>
```

Meta Characters

Meta characters	Description	Example
`.`	match any character except end-of-line (#x0D, #x0A)	`.`
`\`	begin escape sequence	`\n`
`?`	zero or one occurrences	`a?`
`*`	zero or more occurrences	`b*`
`+`	one or more occurrences	`c+`
`{ }`	enclose a numeric quantifier or character group	`{2}`
`( )`	enclose a regular expresion (may be an atom of another regex)	`(abc)`
`[ ]`	enclose a character class expression	`[0-9]`

Regex Quantifiers

Quantifier	Description	Example
`?`	zero or one occurrences	`a?`
`*`	zero or more occurrences	`b*`
`+`	one or more occurrences	`c+`
`{min,max}`	at least ***min*** occurrences, at most ***max*** occurrences	`{2,5}`
`{equ}`	exactly ***equ*** occurrences	`{3}`
`{min,}`	at least ***min*** occurrences	`{2,}`
`{0,max}`	from zero to no more that ***max*** occurrences	`{0,6}`
`{0,0}`	exactly zero occurrences	`{0,0}`

Exemples de pattern

Expression	Match(es)
`Chapter \d`	`Chapter 0, Chapter 1, Chapter 2, ...`
`a*x`	`x, ax, aax, aaax, ...`
`a?x`	`ax, x`
`a+x`	`ax, aax, aaax, ...`
`(a|b)+x`	`ax, bx, aax, abx, bax, bbx, aaax, aabx, abax, abbx, baax, babx, bbax, bbbx, aaaax, ...`
`[abcde]x`	`ax, bx, cx, dx, ex`
`[a-e]x`	`ax, bx, cx, dx, ex`
`[-ae]x`	`-x, ax, ex`
`[ae-]x`	`ax, ex, -x`
`[a-e-[bd]]x`	`ax, cx, ex`
`[^0-9]x`	`Nondigit character followed by the character x`
`\Dx`	`Nondigit character followed by the character x`

```
<xs:simpleType name="orderidtype">
 <xs:restriction base="xs:string">
  <xs:pattern value="[0-9]{6}"/>
 </xs:restriction>
</xs:simpleType>
```

Définition de listes de valeurs

- Permet de définir un élément ayant plusieurs valeurs comme contenu
- Elément <xs:list>

```
<xs:element name=”sizes">
    <xs:simpleType>
        <xs:list itemType="xs:decimal"/>
    </xs:simpleType>
</xs:element>
```

Exemple: `<sizes> 5.5 6.5 7.0 10.5 9.0 4.6 12.8 </sizes>`

Définition de types complexes

- Un « complexType » définit des éléments structurés
  - Un élément structuré contient des éléments ou des attributs
- Définition de types complexes anonymes

```
<?xml version = "1.0"?>
<xs:schema xmlns:xs = "http://www.w3.org/2001/XMLSchema">
   <xs:element name = "Customer">
      <xs:complexType>
         <xs:sequence>
            <xs:element name = "FirstName" type = "xs:string"/>
            <xs:element name = "MiddleName" type = "xs:string"/>
            <xs:element name = "LastName" type = "xs:string"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

Définition de types complexes

- Définition de types complexes nommés

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

  <xs:complexType name = "nameType">
    <xs:sequence>
       <xs:element name = "FirstName" type = "xs:string"/>
       <xs:element name = "MiddleName" type = "xs:string"/>
       <xs:element name = "LastName" type = "xs:string"/>
    </xs:sequence>
  </xs:complexType>

  <xs:element name = "Customer" type = "nameType"/>

</xs:schema>
```

Les compositors

- Un type complexe **ne peut pas** contenir de déclarations d'éléments, mais il peut contenir des "compositors" qui eux contiennent des déclarations d'éléments

- Les compositors permettent de définir comment devront être utilisés les éléments qui les composent

- Il existe 3 genres de compositors:
  - La séquence (sequence)
  - Le choix (choice)
  - « Tous » (all)

La séquence (sequence)

- Indique que tous les éléments figurant à l'intérieur de ce compositor doivent apparaître dans l'ordre dans lequel ils ont été déclarés.

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

```
<?xml version = "1.0"?>
<incomingEmail>
  <from>john.milton@company.com</from>
  <subject>XML Schema workshop</subject>
  <message>Enjoy the workshop!</message>
</incomingEmail>
```

XML DTD (rappel) :

<!ELEMENT incomingEmail (from, subject, message)>

Le choix (choice)

- Défini un choix exclusif entre les éléments qui le composent

```
<?xml version = ”1.0”?>
<xs:schema xmlns:xs = ”http://www.w3.org/2001/XMLSchema”>
  <xs:element name = ”incomingEmail”>
    <xs:complexType>
      <xs:choice>
        <xs:element name = ”from” type = ”xs:string”/>
        <xs:element name = ”subject” type = ”xs:string”/>
        <xs:element name = ”message” type = ”xs:string”/>
      </xs:choice>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

```
<?xml version = ”1.0”?>
<incomingEmail>
  <message>Enjoy the workshop!</message>
</incomingEmail>
```

XML DTD (rappel) :

<!ELEMENT incomingEmail (from | subject | message)>

Tous (`all`)

- Permet aux éléments d'apparaître dans n'importe quel ordre.

```
<?xml version = "1.0"?>
<xs:schema xmlns:xs = "http://www.w3.org/2001/XMLSchema">
  <xs:element name = "incomingEmail">
    <xs:complexType>
      <xs:all>
        <xs:element name = "from" type = "xs:string"/>
        <xs:element name = "subject" type = "xs:string"/>
        <xs:element name = "message" type = "xs:string" minOccurs= "0"/>
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

```
<?xml version = "1.0"?>
<incomingEmail>
  <subject>XML Schema workshop</subject>
  <from>john.milton@company.com</from>
</incomingEmail>
```

XML DTD (rappel) :

<!ELEMENT incomingEmail ((from, subject, message) | (subject, from, message) | …)>

Combinaison de compositors

- Il est possible de combiner `sequence` et `choice`
- Il est interdit de combiner `all` avec d'autres compositors

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

```
<?xml version = "1.0"?>
<Email>
  <from>john.milton@company.com</from>
  <subject>XML Schema workshop</subject>
  <message>Enjoy the workshop!</message>
</Email>
```

Répétitions

- Possibilité d'utiliser `minOccurs` et `maxOccurs` pour définir le nombre d'occurrences des éléments

```
<?xml version = ”1.0”?>
<xs:schema xmlns:xs = ”http://www.w3.org/2001/XMLSchema”>
  <xs:element name = ”Customers”>
    <xs:complexType>
      <xs:sequence minOccurs="0" maxOccurs="unbounded">
        <xs:element name = ”FirstName” type = ”xs:string”/>
        <xs:element name = ”LastName” type = ”xs:string”/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

```
<?xml version = ”1.0”?>
<Customers>
  <FirstName> John </FirstName>
  <LastName> Milton </LastName>
  <FirstName> Joe </FirstName>
  <LastName> Bloggs </LastName>
  …
</Customers>
```

Groupes d'éléments

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

  <xs:group name = "NameGroup">
    <xs:sequence>
      <xs:element name = "FirstName" type = "xs:string"/>
      <xs:element name = "LastName" type = "xs:string"/>
    </xs:sequence>
  </xs:group>

  <xs:element name = "Customers">
    <xs:complexType>
      <xs:group ref="NameGroup" minOccurs = "0" maxOccurs = "unbounded"/>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

Groupes d'attributs

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

 <xs:attributeGroup name = "custLengthGroup">
  <xs:attribute name = "noCustomers" type = "xs:integer"/>
  <xs:attribute name = "Customers" type = "xs:integer"/>
 </xs:attributeGroup>

 <xs:group name = "NameGroup">
  <xs:sequence>
   <xs:element name = "FirstName" type = "xs:string"/>
   <xs:element name = "LastName" type = "xs:string"/>
  </xs:sequence>
 </xs:group>

 <xs:element name = "Customers">
  <xs:complexType>
   <xs:group ref = "NameGroup" minOccurs = "0" maxOccurs = "unbounded"/>
   <xs:attributeGroup ref = "custLengthGroup"/>
  </xs:complexType>
 </xs:element>

</xs:schema>
```

Les modèles de contenu

- Les schémas XML permettent la définition des cinq modèles de contenu suivant:
  - text-only content
  - element-only content
  - mixed content
  - empty content
  - any content

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

  <!- Element-only content -->
  <xs:element name="toto">
    <xs:complexType>
      <xs:sequence>
        <!- Text-only content -->
        <xs:element name="a" type="xs:string"/>
        <xs:element name="b" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>


</xs:schema>
```

Le modèle de contenu mixte (`mixed`)

- Un tel élément peut contenir du texte et des éléments fils

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

 <xs:element name = ”Chapter”>
  <xs:complexType mixed = "true">
   <xs:sequence>
    <xs:element name = "Paragraph" type = "xs:string"/>
    <xs:element name = "Abstract" type = "xs:string"/>
    <xs:element name = "Heading" type = "xs:string"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>

</xs:schema>
```

```
<Chapter>
    The chapter contains the paragraph
    <Paragraph> paragraph1</Paragraph>,
    the abstract
    <Abstract> abstractA </Abstract>
    and has the following heading
    <Heading> heading1 </Heading>
</Chapter>
```

XML DTD (rappel) :

```
<!ELEMENT Chapter (#PCDATA | Paragraph | Abstract | Heading)*>
```

Le modèle de contenu vide

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

  <xs:element name = ”PlaceHolder”>
    <xs:complexType />
  </xs:element>

</xs:schema>
```

sans attribut

Example:

```
<PlaceHolder/>
```

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

  <xs:element name = ”PlaceHolder”>
    <xs:complexType>
      <xs:attribute name = "placeRef" type = "xs:integer"/>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

avec attribut

Example:

```
<PlaceHolder placeRef = “2“/>
```

XML DTD (rappel) :

<!ELEMENT PlaceHolder EMPTY>

Le modèle de contenu (any)

- Un tel élément peut avoir n'importe quel type de contenu

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

 <xs:element name = ”Description”>
  <xs:complexType mixed = "true">
    <xs:sequence>

     <xs:any minOccurs = "0" maxOccurs = "unbounded" processContents = "skip"/>

    <xs:sequence>
  </xs:complexType>
 </xs:element>

</xs:schema>
```

```
<Description>
    Any description possible, e.g.
    <ElementA>Testdata</ElementA> or
    <ElementB Test ="true"/> </Description>
</Description>
```

XML DTD (rappel) :

```
<!ELEMENT Description ANY>
```

En resumé.... la syntaxe des définitions

```
<xs:complexType name = "type_name">
    Compositor
        Déclarations d'éléments
    Déclarations d'attributs
</xs:complexType>
```

```
<xs:simpleType name = "type_name">
    Restrictions de valeurs
    Listes de valeurs
    Unions de types simples
</xs:simpleType>
```

En résumé …

- XML Schéma est un standard défini par le W3C

- Les XML Schémas remplissent la même fonction que les DTD tout en améliorant des aspects comme :
  - Le typage des données
  - La définition des occurrences des éléments
  - Les possibilités de réutilisations de composants du modèle

- Les XML Schémas permettent de rapprocher le monde du XML de celui des bases de données

Quelques références

- Consortium W3C
  - https://www.w3.org/XML/Schema
- W3 Schools
  - https://www.w3schools.com/xml/schema_intro.asp
- Initiation aux Schéma XML
  - http://www.gchagnon.fr/cours/xml/schema.html