مدرسة علوم المعلومات
ⵜⵉⵎⵔⵔⴰⵜ ⵏ ⵜⵓⵙⵙⵏⵉⵡⵉⵏ ⵏ ⵜⵣⵔⴼⵜ
ECOLE DES SCIENCES
DE L'INFORMATION

Elément 02 :

Documents structurés

Pr . J. IDRAIS

Chapitre 3
XML Schema

DTD versus XMLSchema

Les DTD sont souvent suffisantes dans une optique documentation car elles **contrôlent la structure des documents**, elles sont insuffisantes dans une optique de traitement d'information car elles **offrent peu de contrôle sur les types et les nombres d'occurrence**.

- Il est impossible de spécifier un type pour les contenus textuels d'éléments. Tout contenu textuel est #PCDATA que ce soit un entier, une date ou un texte de plusieurs phrases.
- Le typage des valeurs d'attributs est faible (ID, IDREF, énumérations).
- Le contrôle sur le nombre d'occurrences des éléments d'un contenu se résume à 0, 1 ou beaucoup.

DTD versus XMLSchema

Le langage XMLSchema dispose d'un système de typage complet.

- En plus du contrôle de la structure des documents, XMLSchema présente un certain nombre d'avantages qui manquent aux DTD :
- On peut contrôler précisément le nombre d'occurrences d'éléments ou de groupes d'éléments.
- Contrairement aux DTD composées de listes de définitions d'éléments, d'attributs et d'entités, les schémas XMLSchema sont des documents XML bien formés. Ce sont donc des documents XML qui décrivent d'autres documents XML.

XMLSchema
définitions et objectifs

***Définition* Schéma XML**

Un schéma d'un document XML définit :

- les éléments possibles dans un document XML,
- les attributs associés à ces éléments,
- la structure du document et les types de données.

Objectifs des schémas XML

Le langage XML Schema est plus riche et complet que le formalisme des DTD. Il reprend les acquis des DTD . Ainsi, on peut citer quelques objectifs importants :

- Permettre de typer les données,
- Permettre de définir des contraintes sur les types de données ,
- S'intégrer à la galaxie XML.

XMLSchema
définitions et objectifs

Un schéma XML est un document XML avec une extension ".xsd".

- L'élément racine de l'arbre des éléments est " **<xsd :schema>**"
  ou "**<xs :schema>**" ou "**<schema>**" selon l'espace de noms associé.

- Ses éléments de haut niveau servent essentiellement à spécifier :
  - les éléments possibles dans un document XML,
  - les attributs associés à ces éléments,
  - la structure du document et les types de données.

XMLSchema
Structure générale d'un schéma XML

- La structure générale d'un schéma XML est comme suit :

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

XMLSchema
Structure générale d'un schéma XML

```
<!ELEMENT compte (nom, prenom)>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT prenom (#PCDATA)>
```

```
<xsd:schema mlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:element name="info">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="nom" type="xsd:string"/>
            <xsd:element name="prenom" type="xsd:string"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
</xsd:schema>
```

XMLSchema
Spécification d'un élément XML

Les éléments sont déclarés à l'aide de :

- un élément nommé "**xsd :element**" avec un attribut qui donne le nom de l'élément défini.
- Le type de contenu du nouvel élément peut être spécifié par un autre attribut ou par le contenu de la définition de l'élément "**xsd :element**".
- Les déclarations d'éléments peuvent être de deux sortes : **simple** ou **complexe**.

```
<xsd:element name="nom_elt" type="type_elt"/>
```

Exemple :

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

XMLSchema
Spécification d'un élément XML

Modèles de contenu d'un élément XML

Pour un élément XML, différents modèles de contenu existent :

- **Simple** : ne contient que du texte ;
- **complexe** : que des sous-éléments ;
- **mixte** : à la fois du texte et des sous-éléments,
- **vide** : aucun nœud fils.

Dès qu'un élément possède un attribut, il est considéré comme étant **de type complexe, même si son contenu est vide ou simple**.

Le type de contenu d'un élément XML peut être spécifié par l'attribut "***type***" ou par le contenu de la définition de l'élément "**xsd :element**".

XMLSchema
Spécification d'un élément XML

- **Spécification d'élément simple**

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

- **Spécification d'élément complexe**

```
<xsd:element name="point">
     <xsd:complexType>
          <xsd:sequence>
                  <xsd:element name="x" type="xsd:decimal"/>
                  <xsd:element name="y" type="xsd:decimal"/>
          </xsd:sequence>
     </xsd:complexType>
</xsd:element>
```

XMLSchema
Spécification d'un élément XML

Type	Définition
boolean	Valeur booléenne avec true ou 1 pour vrai et false ou 0 pour faux
byte	Nombre entier signé sur 8 bits
unsignedByte	Nombre entier non signé sur 8 bits
short	Nombre entier signé sur 16 bits
unsignedShort	Nombre entier non signé sur 16 bits
int	Nombre entier signé sur 32 bits
unsignedInt	Nombre entier non signé sur 32 bits
long	Nombre entier signé sur 64 bits. Ce type dérive du type xsd:integer.
unsignedLong	Nombre entier non signé sur 64 bits
integer	Nombre entier sans limite de précision. Ce type n'est pas primitif et dérive du type xsd:decimal.
positiveInteger	Nombre entier strictement positif sans limite de précision
negativeInteger	Nombre entier strictement négatif sans limite de précision
nonPositiveInteger	Nombre entier négatif ou nul sans limite de précision
nonNegativeInteger	Nombre entier positif ou nul sans limite de précision
float	Nombre flottant sur 32 bits
double	Nombre flottant sur 64 bits
decimal	Nombre décimal sans limite de précision

- **Types simples prédéfinis**

Les types simples prédéfinis par le langage sont :

- Types numériques
- Types pour les chaînes et les noms XML
- Types pour les dates et les heures
- Types hérités des DTD

XMLSchema
Spécification d'un élément XML

xsd:string	Chaîne de caractères composée de caractères Unicode
xsd:normalizedString	Chaîne de caractères normalisée, c'est-à-dire ne contenant pas de tabulation, de saut de ligne ou de retour chariot
xsd:token	Chaîne de caractères normalisée (comme ci-dessus) et ne contenant pas en outre des espaces en début ou en fin ou des espaces consécutifs
xsd:Name	Nom XML
xsd:QName	Nom qualifié
xsd:NCName	Nom non qualifié, c'est-à-dire sans caractère ':'
xsd:language	Code de langue sur deux lettres de la norme ISO 639
xsd:anyURI	Un URI
xsd:base64Binary	Données binaires représentées par une chaîne au format Base 64.
xsd:hexBinary	Données binaires représentées par une chaîne au format Hex.

- **Types simples prédéfinis**

Les types simples prédéfinis par le langage sont :

- Types numériques
- Types pour les chaînes et les noms XML
- Types pour les dates et les heures
- Types hérités des DTD

XMLSchema
Spécification d'un élément XML

xsd:time	Heure au format hh:mm:ss
xsd:date	Date au format YYYY-MM-DD, par exemple 2016-02-14.
xsd:dateTime	Date et heure au format YYYY-MM-DDThh:mm:ss comme 2008-01-16T14:07:23.
xsd:duration	Durée au format PnYnMnDTnHnMnS comme P1Y6M, P1M12DT2H .
xsd:dayTimeDuration	Durée au format PnDTnHnMnS comme P7DT4H3M2S.
xsd:yearMonthDuration	Durée au format PnYnM comme P1Y6M.
xsd:gYear	Année du calendrier grégorien au format YYYY
xsd:gYearMonth	Année et mois du calendrier grégorien au format YYYY-MM
xsd:gMonth	Jour et mois du calendrier grégorien au format MM-DD
xsd:gMonthDay	Jour et mois du calendrier grégorien au format MM-DD
xsd:gDay	Jour (dans le mois) du calendrier grégorien au format DD

- **Types simples prédéfinis**

Les types simples prédéfinis par le langage sont :

- Types numériques
- Types pour les chaînes et les noms XML
- Types pour les dates et les heures
- Types hérités des DTD

XMLSchema
Type simple dérivé

On peut créer de nouveaux types simples en dérivant des types simples existants. Un nouveau type simple peut être défini par :

- **Restriction** : sous-ensemble des valeurs d'un type existant
- **liste** : liste de valeurs d'un type existant.
- **union** : choix de valeurs parmi un ensemble de types existant.
- **Extension**

Un nouveau type simple est déclaré par un élément "**simpleType**" qui contient un élément : **restriction**, **extension**, **list** ou **union**

La déclaration d'un type simple a la forme suivante :

```
<xsd:simpleType name="name_type"> ... </xsd:simpleType>
```

XMLSchema
Type simple dérivé

Une restriction d'un type simple a deux aspects fondamentaux :

- Nommer le type de base, un type simple existant.
- Décrire les restrictions à l'aide de facettes.

Les Facettes sont des éléments XML Schema dont l'attribut de valeur définit la restriction indiquée par la facette. XML Schema a douze facettes pour restreindre les types simples.

Facette	Applicable au type
minInclusive	nombres et dates
maxInclusive	nombres et dates
minExclusive	nombres et dates
maxExclusive	nombres et dates
totalDigits	nombres
fractionDigits	nombres
length	string, list
minLength	string, list
maxLength	string, list
enumeration	majorité des types
pattern	majorité des types
whitespace	majorité des types

XMLSchema
Type simple dérivé

Une restriction d'un élément avec une valeur entière

```
<xsd:simpleType name="myInteger">
        <xsd:restriction base="xsd:integer">
                <xsd:minInclusive value="10000"/>
                <xsd:maxInclusive value="99999"/>
        </xsd:restriction>
</xsd:simpleType>
```

Une restriction d'un élément email

```
<xsd:simpleType name="Email">
          <xsd:restriction base="xsd:string">
                  <xsd:pattern value="(.)+@(.)+"/>
          </xsd:restriction>
</xsd:simpleType>
```

XMLSchema
Type simple dérivé

Une restriction d'un élément avec une valeur string (ex Password)

```
<xsd:simpleType name="passwordType">
        <xsd:restriction base="xsd:string">
            <xsd:minLength value="8"/>
            <xsd:maxLength value="12"/>
        </xsd:restriction>
</xsd:simpleType>
```

XMLSchema
Définition d'attribut

Un attribut sert comme moyen de structure une information qu'on associera à élément XML.

Dans un schéma XML, il est spécifié comme suit :

```
<xsd:attribute name="nom_attr" type="type_attr"/>
```

```
<xsd:attribute name="nom_attr"  type="type_attr"
                                use="required | optional | prohibited"
                                default="valeur_defaut"
                                fixed="val_fixe"/>
```

L'élément "**xsd :attribute**" est toujours inclus dans un élément "**xsd :complexType**" d'un schéma XML

XMLSchema
Définition du type complexe

Le modèle de contenu des éléments XML complexes, est spécifié par l'élément **complexType** qui peut contenir :

- Des déclarations d'éléments XML fils par :
  - Déclaration directe : spécification de l'élément XML
  - Déclaration par référence : faire référence à une spécification d'élément XML déjà défini (attribut ref).

- Des déclarations d'attributs par l'élément attribute.

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

    <!-- Définition du modèle de contenu -->

</xsd:complexType>
```

XMLSchema
Définition du type complexe

Constructeurs du modèle de contenu complexe

Trois éléments pour exprimer les modèles de contenu :

- Séquence (**sequence**) : les éléments déclarés par les fils doivent apparaître dans l'instance dans le même ordre que dans le schéma.
- Ensemble sans ordre (**all**) : les éléments déclarés par les fils peuvent apparaître au plus une fois dans l'instance, dans un ordre quelconque. all doit être le seul fils d'un modèle de contenu
- Choix (**choice**) : un seul élément déclaré par un des fils peut apparaître dans l'instance

XMLSchema
Définition du type complexe

Constructeurs du modèle de contenu complexe

Un élément "group" offre une facilité d'écriture et est équivalent des entités paramètres de XML (interdit dans all).

Les attributs "**minOccurs**" et "**maxOccurs**" s'appliquent aux éléments "**sequence**", "**all**" et "**choice**", et indiquent le nombre d'occurrences des éléments XML :

- **minOccurs** : entier positif ou nul
- **maxOccurs** : entier positif ou nul, **unbounded**

Les valeurs par défaut des deux attributs : 1 (une occurrence obligatoire si les deux attributs sont omis)

XMLSchema
Exemples de spécification d'éléments XML

Stratégies d'organisation d'un schéma XML

Il existe trois manières d'organiser un schéma XML :

1. Définition du type de chaque élément (autre que ceux ayant un type XSD prédéfini) en utilisant un type anonyme défini comme contenu de la définition de l'élément.
2. Définition des types complexes nommés (types simples aussi redéfinis) et les utiliser dans la définition des éléments.
3. Définition d'éléments en suivant la stratégie 1 et/ou la stratégie 2 et privilégier le référencement.

XMLSchema
Exemples de spécification d'éléments XML

Stratégies d'organisation d'un schéma XML

Définition directe

```
<xsd:element name="cours">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="numero" type="xsd:string"/>
            <xsd:element name="nom" type="xsd:string"/>
            <xsd:element name="date" type="xsd:string"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

XMLSchema
Exemples de spécification d'éléments XML

Stratégies d'organisation d'un schéma XML

Référencement

```
<xsd:element name="numero" type="xsd:string"/>
<xsd:element name="nom" type="xs:string"/>
<xsd:element name="date"  type="xsd:string"/>
<xsd:element name="cours">
     <xsd:complexType>
          <xsd:sequence>
               <xsd:element ref="numero" />
               <xsd:element ref="nom" />
               <xsd:element ref="date" />
          </xsd:sequence>
     </xsd:complexType>
</xsd:element>
```

XMLSchema
Exemples de spécification d'éléments XML

Stratégies d'organisation d'un schéma XML

Séquence d'éléments, groupe et choix

```
<xsd:element name="personne" type="typePersonne"/>
<xsd:complexType name="typePersonne">
     <xsd:sequence>
            <xsd:group ref="civilite"/>
            <xsd:element name="dateDeNaissance" type="xsd:date"/>
            <xsd:choice>
                  <xsd:element name="adresse" type="xsd:string"/>
                  <xsd:element name="email" type="xsd:string"/>
            </xsd:choice>
            <xsd:element name="telephone" type="numeroDeTelephone"/>
        </xsd:sequence>
</xsd:complexType>
<xsd:group name="civilite">
      <xsd:sequence>
              <xsd:element name="nom" type="xsd:string"/>
              <xsd:element name="prenom" type="xsd:string"/>
      </xsd:sequence>
</xsd:group>
```

XMLSchema
Exemples de spécification d'éléments XML

Stratégies d'organisation d'un schéma XML

Cas d'attribut

```
<?xml version="1.0" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema">
      <xsd:element name="contacts" type="typeContacts"/>
      <xsd:element name="remarque" type="xsd:string"/>
      <xsd:complexType name="typeContacts">
            <!-- content declaration-->
            <xsd:attribute name="maj" type="xsd:date"/>
      </xsd:complexType>
</xsd:schema>
```

XMLSchema
Exemples de spécification d'éléments XML

Stratégies d'organisation d'un schéma XML

Groupe d'attributs

```
<?xml version="1.0" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema">
            <xsd:element name="contacts" type="typeContacts"/>
            <xsd:element name="remarque" type="xsd:string "/>
            <xsd:complexType name="typeContacts ">
                  <xsd:attributeGroup ref="InfosMaj"/>
            </xsd:complexType>
            <xsd:attributeGroup name="InfosMaj">
                   <xsd:attribute name="maj" type="xsd:date"/>
                   <xsd:attribute name="auteur" type="xsd:string"/>
            </xsd:attributeGroup>
</xsd:schema>
```

XMLSchema
Exemples de spécification d'éléments XML

Stratégies d'organisation d'un schéma XML

- Constructeur All

```
<?xml version="1.0" ?>
<xsd:complexType name="typePersonne">
   <xsd:all>
            <xsd:element name="nom" type="xsd:string"/>
            <xsd:element name="prenom" type="xsd:string"/>
            <xsd:element name="dateDeNaissance" type="xsd:date"/>
            <xsd:element name="adresse" type="xsd:string"/>
            <xsd:element name="email" type="xsd:string"/>
            <xsd:element name="telephone" type="xsd:string"/>
   </xsd:all>
</xsd:complexType>
```