Données semi-structurées

I. XML

1. Introduction

Historique

XML semble d'être comme HTML, mais totalement différents. Tous deux sont les successeurs de SGML, lui-même issu de GML. Ce dernier a été conçu par IBM pour dissocier l'apparence des documents textes de leur contenu, en particulier des documentations techniques. Il est facile de changer l'apparence (mise en page, polices, couleurs. . .) d'un tel document sans en réécrire une seule ligne. D'autres langages, comme LaTeX sont similaires: on écrit le texte sans se soucier de la mise en page. Cela permet également de le traduire dans d'autres langues. Inversement, les logiciels *wysiwyg* comme Word mélangent mise en page et contenu. Même avec des styles, il reste difficile de modifier l'apparence d'un document sans devoir le réécrire au moins partiellement.

XML peut être vu comme une généralisation de SGML et HTML permettant de construire toutes sortes de documents.

XML (« Extensible Markup Language ») ?

C'est un langage permettant de représenter et structurer des informations à l'aide de balises (« tag »). Chacun peut définir et employer les balises comme il le veut.

Exemple : texte ... <BALISE> ... texte … </BALISE> ...

Pourquoi XML ?

Le format XML est au cœur de nombreux processus actuels :
• échange de données entre serveurs et clients,
• outils et langages de programmation,
• bases de données XML natives.
XML définit la syntaxe des documents, mais les applications définissent les balises, leur signification et ce qu'elles doivent contenir. Des API existent dans tous les langages pour lire et écrire des documents XML.

Exemple de fichier XML

Un fichier XML représente des informations structurées (bien typé). L'exemple suivant modélise un itinéraire composé d'étapes :

```
<?xml version="1.0" encoding="utf-8"?>
<!-- itinéraire fictif -->
<itineraire>
   <etape distance="0km">départ</etape>
   <etape distance="13km">tourner à droite</etape>
   <etape distance="22km">arrivée</etape>
</itineraire>
```

On choisit les balises et les attributs comme on le souhaite. Il faut seulement que ça soit homogène, régulier afin de pouvoir être exploité par un logiciel (parseur).

On peut faire ceci, mais bonjour les difficultés pour le traiter :

```
<?xml version="1.0" encoding="utf-8"?>
<!-- itinéraire fictif -->
<itineraire>
<etape distance="0km">départ</etape>
<Etape><distance>13km</distance>tourner à droite</Etape>
<etape distance="22km"><infos>arrivée</infos></etape>
</itineraire>
```

Exemple d'application de XML

XML en tant que format de fichier :
Plusieurs outils informatiques utilisent directement un Format XML pour enregistrer leurs fichiers :
Bureautique : LibreOffice utilise le format OpenDocument**.**
Dessin vectoriel avec Inkscape, format SVG.
Équations mathématiques, format MathML,
…

Bases de données XML
Bases de données XML native : les données sont au format XML et les requêtes sont dans un langage (XQuery) permettant de réaliser l'équivalent de SQL.

Échange de données entre clients et serveur
XML est le format utilisé pour représenter des données volatiles de nombreux protocoles comme RSS, XML-RPC, SOAP , AJAX, …

2. Structure d'un document XML

Un document XML est composé de plusieurs parties :

a) Entête de document précisant la version et l'encodage,
b) Des règles optionnelles permettant de vérifier si le document est valide « Document type ».
c) Un arbre d'éléments partant d'un élément appelé racine.
   – Un élément possède un nom, des attributs et un contenu
   – Le contenu d'un élément peut être :
      ∗ rien : élément vide noté **<nom/>** ou **<nom attributs.../>**
      ∗ du texte

∗ d'autres éléments (les éléments enfants).

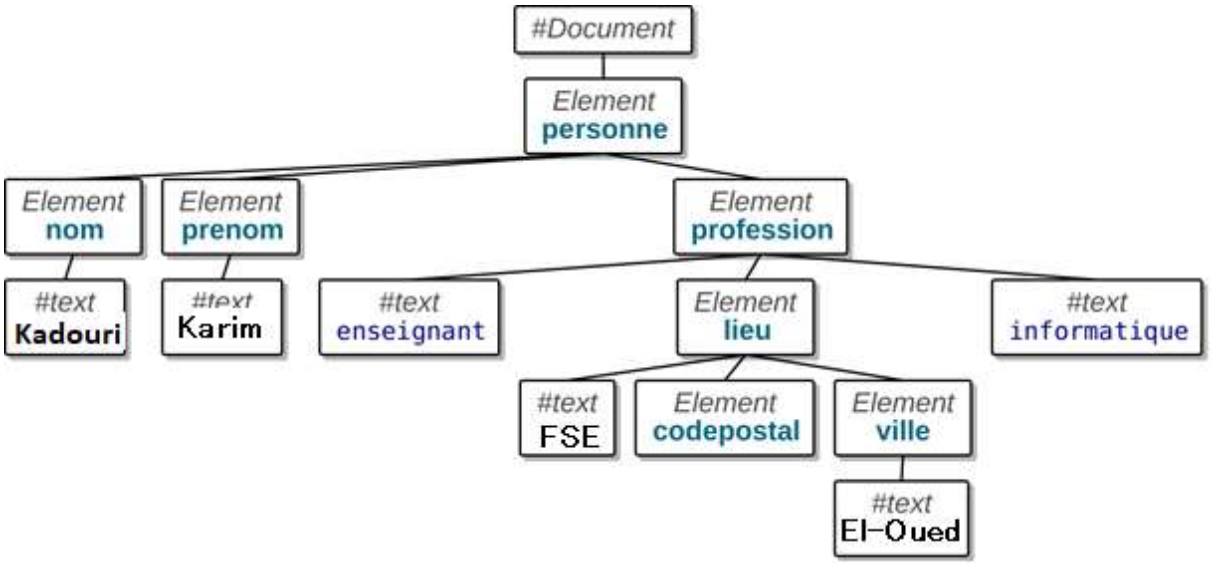
– Un élément non vide est délimité par une balise ouvrante et une balise fermante.

∗ une balise ouvrante est notée <nom attributs...>

∗ une balise fermante est notée </nom>

Voici l'exemple un document XML représentant une personne :

```
<?xml version="1.0" encoding="utf-8"?>
<personne>
      <nom>Kadouri</nom>
      <prenom>Karim</prenom>
      <profession>  enseignan
            <lieu>
                Faculté Sciences Exactes
                <codepostal/>
                <ville>El-Oud</ville>
            </lieu>
            informatique
      </profession>
</personne>
```



Prologue XML

La première ligne d'un document XML est appelée prologue XML. Elle spécifie la version de la norme XML utilisée (1.0 ou 1.1 qui sont très similaires) ainsi que l'encodage :

```
<?xml version="1.0" encoding="utf-8"?>
<?xml version="1.0" encoding="iso-8859-15"?>
```

Il y a plusieurs normes d'encodage :

• ASCII ne représente que les 128 premiers caractères Unicode.
• ISO 8859-1 ou Latin-1 représente 191 caractères de l'alphabet latin n°1 (ouest de l'Europe) à l'aide d'un seul octet. Les caractères € et œ n'en font pas partie, pourtant il y a æ.
• ISO 8859-15 est une norme mieux adaptée au français. Elle rajoute € et œ et supprime des caractères peu utiles comme ¤.
• UTF-8 représente les caractères à l'aide d'un nombre variable d'octets, de 1 à 4 selon le caractère.
Par exemple : A est codé par 0x41, é par 0xC3,0xA9 et € par 0xE2,0x82,0xAC.

Commentaire XML

Voici un exemple d'un commentaire XML

```
<!-- voici un commentaire -->

<!--
voici un autre commentaire
et ça -- , c'est une erreur
-->
```

La seule contrainte est de ne pas pouvoir employer les caractères -- dans le commentaire, même s'ils ne sont pas suivis de >.

Après le prologue, on peut trouver plusieurs parties optionnelles délimitées par <?...?> ou <!...>. Ce sont des instructions de traitement, des directives pour l'analyseur XML.
Par exemple :
• un *Document Type Definitions* (DTD) qui permet de valider le contenu du document.
• une feuille de style,

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE personne SYSTEM "personne.dtd">
<?xml-stylesheet href="personne.css" type="text/css"?>
<personne>
...
</personne>
```

2.1 Éléments XML

Un élément XML est délimité par :
• une balise ouvrante <nom attributs...>
• une balise fermante </nom> obligatoire.
Le contenu de l'élément se trouve entre les deux balises. Ce sont des éléments enfants et/ou du texte.

```
<parent>
  texte1
  <enfant>texte2</enfant>
  texte3
</parent>
```

Dans le cas où l'élément n'a pas de contenu (*élément vide*), on peut regrouper ses deux balises en une seule <nom attributs.../>

Le même type d'élément peut se trouver plusieurs fois avec le même parent, avec des contenus identiques ou différents :

```
<element1>
  <element2>texte1</element2>
  <element2>texte2</element2>
  <element2>texte1</element2>
</element1>
```

Le caractère : permet de séparer le nom en deux parties : préfixe et *nom local*. Le tout s'appelle *nom qualifié*. Par exemple iut:departement est un nom qualifié préfixé par iut. Ce préfixe permet de définir un *espace de nommage.*

nom qualifié = préfixe:*nom local*

Espaces de nommage

Un espace de nommage (namespace) définit une famille de noms afin d'éviter les confusions entre des éléments qui auraient le même nom mais pas le même sens. Cela arrive quand le document XML modélise les informations de plusieurs domaines.
Exemple avec confusion :

```
<meuble id="765">
 <table prix="74,99€">Produit</table>
   <table border="1">
   <tr><th>longueur</th><th>largeur</th></tr>
   <tr><td>120cm</td><td>80cm</td></tr>
 </table>
</meuble>
```

Voici maintenant l'exemple précédent avec des espaces de nommage :

```
<?xml version="1.0" encoding="utf-8"?>
    <meuble:meuble id="765"
       xmlns:meuble="urn:iutlan:meubles"
       xmlns:html="http://www.w3.org">

    <meuble:table prix="74,99€"> acajou </meuble:table>

    <html:table border="1">
      <html:tr><html:th>longueur</html:th>...</html:tr>
      <html:tr><html:td>120cm</html:td>...</html:tr>
    </html:table>

</meuble:meuble>
```

Lorsque la racine du document définit un attribut xmlns="URI", alors par défaut toutes les balises du document sont placées dans ce namespace, donc pas besoin de mettre un préfixe. Ça peut s'appliquer également localement à un élément et ça concerne aussi toute sa descendance:

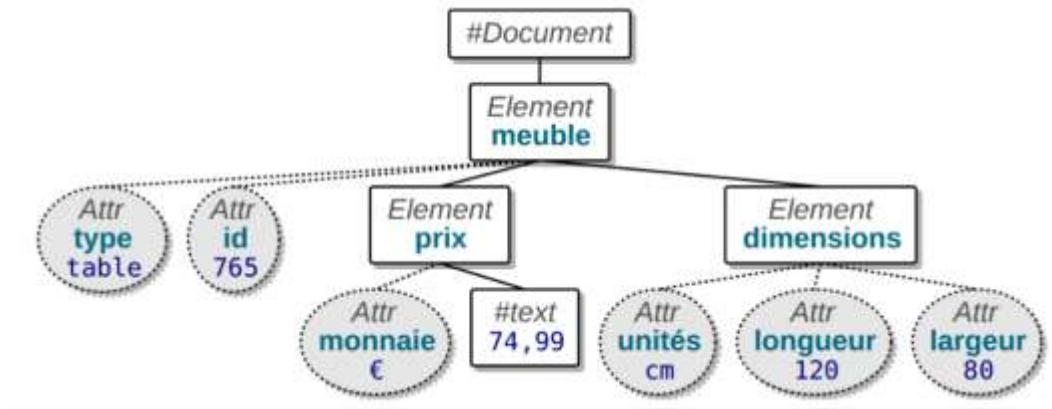
```
<meuble id="765" xmlns="urn:iutlan:meubles">
  <table prix="74,99€">acajou</table>
   <table border="1" xmlns="http://www.w3.org">
       <tr><th>longueur</th>...</tr>
      <tr><td>120cm</td>...</tr>
  </table>
</meuble>
```

Attributs

Les attributs caractérisent un élément. Ce sont des couples nom="valeur" ou nom='valeur'. Ils sont placés dans la balise ouvrante.

Exemple :

```
<?xml version="1.0" encoding="utf-8"?>
  <meuble id="765" type='table'>
    <prix monnaie='€'>74,99</prix>
    <dimensions unites="cm" longueur="120" largeur="80"/>
   <description langue='fr'>Belle petite table</description>
</meuble>
```

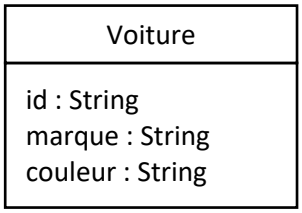


Remarques :

Il n'y a pas d'ordre entre les attributs d'un élément. Un attribut ne peut être présent qu'une fois.

3. Modélisation des tables (Entités)

Soit la table voiture suivante :



Voici des représentations XML possibles pour cette table :

<?xml version="1.0" encoding="utf-8"?> <voitures> <voiture id="125" marque="Renault" couleur="grise"/> <voiture id="982" marque="Peugeot" couleur="noire"/> </voitures>	<?xml version="1.0" encoding="utf-8"?> <voitures> <voiture> <id>125</id> <marque>Renault</marque> <couleur>grise</couleur> </voiture> <voiture> <id>982</id> <marque> Peugeot </marque> <couleur> noire </couleur> </voiture> </voitures>

Attributs ou sous-éléments ?

Dans la réflexion pour choisir de représenter une information sous forme d'attributs ou de sous-éléments,
il faut prendre en compte ces critères :

• Les attributs ne peuvent pas contenir plusieurs valeurs, tandis qu'on peut avoir plusieurs sous-éléments ayant le même nom et des contenus différents,

• Les attributs ne peuvent pas contenir de structures d'arbres, au contraire des éléments,

• Les attributs ne sont pas facilement extensibles, tandis qu'on peut toujours rajouter de nouveaux sous-éléments dans une hiérarchie sans changer les logiciels existants,

• Il n'est pas plus compliqué d'accéder à des sous-éléments qu'à des attributs car tous sont des nœuds dans l'arbre sous-jacent

4 Validation d'un document XML

Pourquoi la validation :
Échange de données, imaginez qu'une personne X veut échanger de données concernant les étudiants : nom, prénom, date naissance, moyen, … . Il faut que les deux correspondants soient d'accord sur un format de données ??

Les DTD (Document Type Definition), étant la forme la plus ancienne, sont présentes dans la plupart des outils. Viennent ensuite ce qu'on nomme les schémas W3C, une forme de grammaire plus moderne mais également plus complexe.

4.1 Validation par DTD

Une DTD (Document Type Definition) est une forme de grammaire relativement ancienne car issue de l'univers SGML (Standard Generalized Markup Language).

Par exemple, un document XML ayant une DTD externe cours.dtd, située dans le même répertoire que notre document XML (accès relatif), se présente sous la forme :

```
<?xml version="1.0"?>
<!DOCTYPE cours SYSTEM "cours.dtd">
<cours>
…
</cours>
```

Dans cet exemple, la DTD cours.dtd est localisée relativement à notre document XML. Le mot-clé SYSTEM est important et indique qu'il s'agit d'une DTD qui vous est propre. L'alternative est le mot-clé PUBLIC.

La définition d'un élément

```
<!ELEMENT NomElemnt DEF_CONTENU>
```

DEF_CONTENU peut contenir :

• EMPTY : l'élément n'a pas de contenu ; il est donc vide. Il peut cependant avoir des attributs.

• ANY : l'élément peut contenir n'importe quel élément présent dans la DTD.

• (#PCDATA) : l'élément contient du texte. Le caractère # est là pour éviter toute ambiguïté avec une balise et indique au parseur qu'il s'agit d'un mot-clé. PCDATA signifie *Parsable Character DATA*.

• Un élément placé entre parenthèses comme (nom_element). Le nom d'un élément désigne une référence vers un élément décrit dans une autre partie de la DTD

• Un ensemble d'éléments séparés par des opérateurs, le tout placé entre parenthèses.

L'opérateur de choix, représenté par le caractère **"|",** indique que l'un ou l'autre de deux éléments (ou deux ensembles d'éléments) doit être présent. L'opérateur de suite (ou séquence), représenté par le caractère **","**, indique que les deux éléments (ou les deux ensembles d'éléments) doivent être présents. Des parenthèses supplémentaires peuvent être utilisées pour lever les ambiguïtés

Quelques exemples :

```
<!ELEMENT personne (nom_prénom | nom)>
```

```
<!ELEMENT nom_prenom (#PCDATA)>
<!ELEMENT nom (#PCDATA)>
```

C'est la forme d'un grammaire :

Personne → nom_prenom | nom

nom_prenom → text

nom → text

Cela nous autorise deux documents XML, soit :

```
<personne>
  <nom_prenom>Karim Kadouri</nom_prenom>
</personne>
```

ou bien :

```
<personne>
  <nom>Kadouri</nom>
</personne>
```

Autre cas avec l'opérateur de séquence.

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

C'est comme un grammaire :

```
Personne → prenom nom
Prenom → text
Nom → text
```

Ici, l'opérateur de séquence limite les possibilités à un seul document XML valide :

```
<personne>
<prenom>Karim</prenom>
<nom>Kadouri</nom>
</personne>
```

~~<personne>~~
~~<nom>Kadouri</prenom>~~
~~<prenom>Karim</nom>~~
~~</personne>~~

Les contenus (élément ou groupe d'éléments) peuvent être quantifiés par les opérateurs *, + et ?. Ces opérateurs sont liés au concept de cardinalité. Lorsqu'il n'y a pas d'opérateur, la quantification est de 1 (donc toujours présent).
Voici le détail de ces opérateurs :

• * : 0 à n fois ;

• + : 1 à n fois ;
• ? : 0 ou 1 fois.

Quelques exemples :

```
<!ELEMENT plan (introduction?,chapitre+,conclusion?)>
```

L'élément plan contient un élément introduction optionnel, suivi d'au moins un élément chapitre et se termine par un élément conclusion optionnel également.

```
<!ELEMENT chapitre (auteur*,paragraphe+)>
```
L'élément chapitre contient de 0 à n éléments auteur suivi d'au moins un élément paragraphe.

```
<!ELEMENT livre (auteur?,chapitre)+>
```
L'élément livre contient au moins un élément, chaque élément, étant un groupe d'éléments où l'élément auteur, est optionnel et l'élément chapitre est présent en un seul exemplaire.

Définition d'attributs

Les attributs sont précisés dans l'instruction **ATTLIST**. Cette dernière, étant indépendante de l'instruction **ELEMENT**, on précise à nouveau le nom de l'élément sur lequel s'appliquent le ou les attributs. On peut considérer qu'il existe cette forme syntaxique :

```
< ATTLIST  NOM_ELEMEMENT
  Attr_def* … >
```

Attr_def est détaillée comme suit:

```
nom TYPE OBLIGATION VALEUR_PAR_DEFAUT
```

Le TYPE peut être principalement :

• CDATA : du texte (*Character Data*) ;

• ID : un identifiant unique (combinaison de chiffres et de lettres) ;

• IDREF : une référence vers un ID ;

• IDREFS : une liste de références vers des ID (séparation par un blanc) ;

• NMTOKEN : un mot (donc pas de blanc) ;

• NMTOKENS : une liste de mots (séparation par un blanc) ;

• Une énumération de valeurs : chaque valeur est séparée par le caractère |.

L'OBLIGATION ne concerne pas les énumérations qui sont suivies d'une valeur par défaut.

Dans les autres cas, on l'exprime ainsi :

• #REQUIRED : attribut obligatoire.

• #IMPLIED : attribut optionnel.

• #FIXED : attribut toujours présent avec une valeur. Cela peut servir, par exemple, à

imposer la présence d'un espace de noms.
La VALEUR_PAR_DEFAUT est présente pour l'énumération ou lorsque la valeur est typée avec #IMPLIED ou #FIXED.
Quelques exemples :

```
<!ATTLIST chapitre
  titre CDATA #REQUIRED
  auteur CDATA #IMPLIED >
```

L'élément chapitre possède ici un attribut titre obligatoire et un attribut auteur optionnel.

```
<!ATTLIST crayon
couleur (rouge|vert|bleu) "bleu">
```

L'élément crayon possède un attribut couleur dont les valeurs font partie de l'ensemble rouge, vert, bleu.

Définition d'entités

Les entités sont déclarées par l'instruction ENTITY. Comme nous l'avons abordé dans le chapitre précédent, l'entité associe un nom à une valeur. Ce nom est employé dans le document XML comme une forme d'alias ou de raccourci vers la valeur suivant la syntaxe &nom;. La valeur d'une entité peut être interne ou externe.

La syntaxe pour déclarer une entité est simplement la suivante :

```
<!ENTITY nom "VALEUR">          cas interne.
<!ENTITY nom SYSTEM "unTexte.txt">     cas externe.
```

Exemples :

Exemple1

```
<?xml version="1.0" encoding="utf-8"?>
    <!DOCTYPE itineraire [
```

```
      <!ELEMENT itineraire (boucle?, etape+, variante*)>
    <!ELEMENT boucle EMPTY>
    <!ELEMENT etape (#PCDATA)>
    <!ELEMENT variante ANY>
     ]>
<itineraire>
   <boucle/>
   <etape>départ</etape>
   <etape>tourner à droite</etape>
   <variante>
      <etape>départ</etape><tape>tourner à gauche</tape>
   </variante>
</itineraire>
```

Ex1

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE cours [
                 <!ENTITY auteur "Alexandre Brillant">
                 <!ELEMENT cours (#PCDATA)>
                ]>
<cours>
Cours réalisé par &auteur;
</cours>
```

Ex2

```
<?xml version="1.0" encoding="utf-8"?>
  <!DOCTYPE itineraire [
    <!ELEMENT itineraire (etape+)>
    <!ATTLIST itineraire nom CDATA #IMPLIED>
    <!ELEMENT etape (#PCDATA)>
    <!ATTLIST etape distance CDATA #REQUIRED>
     ]>
 <itineraire nom="essai">
   <etape distance="0km">départ</etape>
   <etape distance="1km">tourner à droite</etape>
 </itineraire>
```

Ex3

```
<!ELEMENT agence (nom, bureau+, periode, voyage*) >
<!ATTLIST agence  niveau CDATA #IMPLIED>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT  bureau (#PCDATA)>
<!ATTLIST bureau code_bureau  ID #REQUIRED>
<!ELEMENT periode (#PCDATA) >
<!ELEMENT voyage (pays, duree, descriptif, prix) >
<!ATTLIST voyage niveau (1 | 2 | 3 | 4 | 5) #IMPLIED
                                responsable IDREF #REQUIRED
                                code_voyage ID #REQUIRED>
 <!ELEMENT descriptif  (#PCDATA | remarque | voir )* >
 <!ATTLIST descriptif  langue CDATA "fr" >
 <!ELEMENT remarque (#PCDATA) >
 <!ELEMENT voir EMPTY >
 <!ATTLIST voir autres IDREFS #REQUIRED>
 <!ELEMENT prix (#PCDATA) >
 <!ELEMENT  pays (#PCDATA) >
 <!ELEMENT duree (#PCDATA) >
```

4.2 XML Schema