

# Données Semi-Structurées:

## XML, XPath, DTD

Enseignant: *Dario COLAZZO*  
Chargée de TD/TP: *Beatrice NAPOLITANO*

18 Janvier 2020

### 1 Rappel

#### Rappel 1 (document XML bien formé) :

*Un document XML (eXtensible Markup Language) bien formé a la bonne syntaxe XML.*

*Les règles de syntaxe sont :*

- *les documents XML doivent avoir un préambule et un seul élément racine*
- *les éléments doivent avoir une balise de fermeture*
- *les éléments doivent être correctement imbriqués*
- *les valeurs d'attribut XML doit être citées*
- *les caractères < et & sont strictement illégaux et doivent être échappés*

*Ces cinq entités sont définies par défaut : <, >, &, ' and ". Ils ont des références d'entités prédéfinies :*

<code>&amp;lt;</code>	<code>&lt;</code>	<i>less than</i>
<code>&amp;gt;</code>	<code>&gt;</code>	<i>greater than</i>
<code>&amp;amp;</code>	<code>&amp;</code>	<i>ampersand</i>
<code>&amp;apos;</code>	<code>'</code>	<i>apostrophe</i>
<code>&amp;quot;</code>	<code>"</code>	<i>quotation mark</i>

#### Rappel 2 (document XML valide) :

*Un document XML est valide quand :*

- *il est bien formé*
- et*
- *il est également conforme aux règles d'une Document Type Definition (DTD).*

*On dit que le document XML est valide par rapport à la DTD. Le DTD peut être externe ou interne.*

#### Rappel 3 (syntaxe DTD) :

*DTD externe :*

```
1: <?xml version="1.0"?>
2: <!DOCTYPE bib SYSTEM "bib.dtd">
3: <bib>
4:   ...
5: </bib>
```

*DTD interne :*

```
1: <?xml version="1.0"?>
2: <!DOCTYPE bib [
3: <!ELEMENT bib (book*)>
4: <!ELEMENT book ...>
5: ...
6: ]>
7: <bib>
```

8: ...  
9: <bib>

#### Rappel 4 (Document Type Definition) :

*DTD : ensemble de declaration spécifiant la structure d'un type d'élément ou d'attribut.*

- 1) *Type élément : tag + structure contenu (expression régulière)*  
`<!ELEMENT tag structure>`

*Expression régulière :*

- *A,B : un A suivi d'un B*
  - *A\* : 0 ou plusieurs A*
  - *A? : 0 ou 1 A*
  - *B+ : 1 ou plusieurs B*
  - *(A|B) : un A ou B dans n'importe quel ordre*
- où A et B peut être autres éléments ou #PCDATA (chaîne de caractère).  
La structure peut aussi être EMPTY soit ANY.*

- 2) *Type attribut : tag + type contenu + contrainte*  
`<!ATTLIST tag_element tag_attribut type_contenu contrainte>`

*Le type contenu peut être :*

- *CDATA*
- *ID*
- *IDREF*
- *s1,...,sn*

*La contrainte peut être :*

- *#REQUIRED*
- *#IMPLIED*
- *#FIXED value*

#### *Example :*

- **#REQUIRED**

*DTD :*

- 1: `<!ELEMENT contact EMPTY>`
- 2: `<!ATTLIST contact number CDATA #REQUIRED>`

*XML :*

- 1: `<contact number="06-12345678" />`

*si nous écrivons*

- 1: `<contact />`

*c'est une erreur.*

- **#IMPLIED**

*DTD :*

- 1: `<!ELEMENT contact EMPTY>`
- 2: `<!ATTLIST contact fax CDATA #IMPLIED>`

*XML :*

- 1: `<contact fax="555-667788" />`

*si nous écrivons*

- 1: `<contact />`

*est correcte et la valeur du champ fax est nulle.*

- **#FIXED value**

*DTD :*

- 1: <!ELEMENT sender EMPTY>
- 2: <!ATTLIST sender person CDATA #FIXED "Beatrice">

XML :

- 1: <sender person="Beatrice" />
- 2: <sender />

sont ok mais si nous écrivons

- 1: <sender person="Dario" />

c'est une erreur.

#### — **default value**

DTD :

- 1: <!ELEMENT square EMPTY>
- 2: <!ATTLIST square width CDATA "0">

XML :

- 1: <square width="100" />

la valeur de width est 100, mais si on ne le spécifie pas

- 1: <square />

c'est correct e la valeur est 0.

### Rappel 5 (Expressions XPath) :

[/]étape /étape /.../étape

Une étape : trois composants `axe::filtre[prédicat_1]...[prédicat_n]`

— *L'axe : sens de parcours des nœuds*

- *ancestor*
- *ancestor-or-self*
- *attribute*
- *child*
- *descendant*
- *descendant-or-self*
- *following*
- *following-sibling*
- *namespace*
- *parent*
- *preceding*
- *preceding-sibling*
- *self*

— *Le filtre : type des nœuds qui seront retenus*

- `child::book` les fils de nœud contexte ayant tag book
- `child::node()` tout les fils du nœud contexte
- `child::*` tout les fils élément du nœud contexte
- `child::text()` tout les fils texte du nœud contexte

— *Le(s) prédicat(s) : propriétés que doivent satisfaire les nœuds retenus par `axe::filtre`*

## 2 Links

Valdateurs en ligne de documents XML :

- [http://www.w3schools.com/xml/xml\\_validator.asp](http://www.w3schools.com/xml/xml_validator.asp)
- <http://www.xmlvalidation.com/>

### 3 Exercice

#### Exercice 1 (XML) :

Décider si les documents XML suivants sont correctes ? Sinon, trouvez les erreurs et les corriger.

1. 

```
1: <?xml version="1.0" encoding="UTF-8"?>
2: <html>
3:   <head><title>Hello, World</title></head>
4:   <body><p>Hello, World</p></body>
5: </html>
```
2. 

```
1: <?xml version="1.0" encoding="UTF-8"?>
2: <p>This is a test. This is a test of the <em>
3:   <strong>Emergency</em> Broadcast System.</strong></p>
```
3. 

```
1: <?xml version="1.0" encoding="UTF-8"?>
2: <note date="12/11/2007">
3:   <!-- This is a comment -->
4:   <Message><to>Tove</to>
5:   <from>Jani</from>
6:   <heading>Reminder</heading>
7:   <body>Don't forget me this weekend!</body></message>
8: </note>
9: <note date="13/11/2007">
10:  <message><to>Jani</to>
11:  <from>Tove</from>
12:  <heading>Re: Reminder</heading>
13:  <body>Ok!</body></message>
14: </note>
```
4. 

```
1: <?xml version="1.0" encoding="utf-8"?>
2: <xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
3: xmlns:xs="http://www.w3.org/2001/XMLSchema">
4:   <xs:element name="points">
5:     <xs:complexType>
6:       <xs:sequence>
7:         <xs:element maxOccurs="unbounded" name="point">
8:           <xs:complexType>
9:             <xs:attribute name="x" type="xs:unsignedShort" use="required" />
10:            <xs:attribute name="y" type="xs:unsignedShort" use="required" />
11:          </xs:complexType>
12:        </xs:element>
13:      </xs:sequence>
14:    </xs:complexType>
15:  </xs:element>
16: </xs:schema>
```
5. 

```
1: <?xml version="1.0" encoding="UTF-8"?>
2: <html>
3: <head><title>Paragraphs</title></head>
4: <body><p>This is a paragraph.<br/>
5: <p>This is another paragraph.<br/>
6: <p>Third paragraph.</body>
7: </html>
```
6. 

```
1: <?xml version="1.0" encoding="UTF-8"?>
```

```

2: <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3:   xmlns:dc="http://purl.org/dc/elements/1.1/">
4:   <rdf:Description rdf:about="http://www.AcronymFinder.com/">
5:     <dc:title>Acronym Finder</dc:title>
6:     <dc:description>The Acronym Finder is a world wide
7:       web (WWW) searchable database of more than 169,000
8:       abbreviations and acronyms about computers,
9:       technology, telecommunications, and military
10:      acronyms and abbreviations.</dc:description>
11:     <dc:subject>
12:       <rdf:Bag>
13:         <rdf:li>Astronomy</rdf:li>
14:         <rdf:li>Literature</rdf:li>
15:         <rdf:li>Mathematics</rdf:li>
16:         <rdf:li>Music</rdf:li>
17:         <rdf:li>Philosophy</rdf:li>
18:       </rdf:Bag>
19:     </dc:subject>
20:   </rdf:Description>
21: </rdf:RDF>

```

7. 1: <?xml version="1.0" encoding="UTF-8"?>  
2: <html><body>  
3: <p><b><i>This paragraph is bold and italic.</b></i></p><br/>  
4: <p><i><b>This paragraph is italic and bold.</i></b></p><br/>  
5: </body></html>

8. 1: <catalog>  
2: <work type='prose' date='1906'>  
3: <title>The Gift Of The Magi</title>  
4: <author>O. Henry</author>  
5: </work>  
6: <work type='poem' date='1845'>  
7: <title>The Raven</title>  
8: <author>Edgar Allen Poe</author>  
9: </work>  
10: <work type='play' date='1601'>  
11: <title>Hamlet</title>  
12: <author>William Shakespeare</author>  
13: </work>  
14: </catalog>

9. 1: <?xml version="1.0" encoding="UTF-8"?>  
2: <letter>  
3: <date>December 11, 2002</date>  
4: <addressee>  
5: <name>Melville Dewey</name>  
6: <address\_one>  
7: Columbia University</address\_one><address\_two>New York, NY  
8: </address\_two>  
9: </addressee>  
10: <greeting>Dear Melville,</greeting>  
11: <paragraph>I have been reading your ideas concerning nature of  
12: librarianship, and <i>I find them very intriguing</i>.  
13: I would love the opportunity to discuss with you the role of the  
14: card catalog in today's libraries considering the advent to World

```

15:     Wide Web. Specifically, how are things like Google and Amazon.com
16:     changing our patrons' expectations of library services? Mr. Cutter
17:     and I will be discussing these ideas at the next Annual Meeting,
18:     and we are available at the follow dates/times:</paragraph>
19:     <list>
20:         <item>Monday, 2-4</item>
21:         <item>Tuesday, 3-5</item>
22:         <item>Thursday, 1-3</item>
23:     </list>
24:     <paragraph>We hope you can join us.</paragraph>
25: <closing>Sincerely, S. R. Ranganathan</closing>
26: </letter>

10. 1: <?xml version="1.0"?>
    2: <dictionary>
    3:   <word>
    4:     <update date="2002-12-23"/>
    5:     <name is_acronym="true">XML</Name>
    6:     <description>eXtensible Markup Language</description>
    7:   </word>
    8:   <word>
    9:     <update date="2002-12-23"/>
    10:    <name is_acronym="true">POP</name>
    11:    <definition default>Post Office Protocol</definition>
    12:    <definition>Point Of Purchase</definition>
    13:  </dictionary>

11. 1: <?xml version="1.0" encoding="UTF-8"?>
    2: <domain type='kvm'>
    3:   <name>domain</name><
    4:     <memory>524288</memory>
    5:     <vcpu>2</vcpu>
    6:     <features><acpi/><pae/>
    7:     <clock offset='utc'>
    8:     <disk type='block' device='cdrom'>
    9:       <driver name='qemu' type='raw'>
    10:       <source file='/path/to/image.iso'>
    11:       <tar get dev='hdc' bus='ide'>
    12:       <readonly/></name>
    13:     </disk>
    14:   </domain>

12. 1: <?xml version="1.0" encoding="UTF-8"?>
    2: <name>Oyster Soup</name>
    3: <author>Eric Lease Morgan</author>
    4: <copyright holder=Eric Lease Morgan>&copy; 2003</copyright>
    5: <ingredients>
    6:   <list>
    7:     <item>1 stalk of celery
    8:     <item>1 onion
    9:     <item>2 tablespoons of butter
    10:    <item>2 cups of oysters and their liquor
    11:    <item>2 cups of half & half
    12:   </list><cost>total cost < 36 euro </cost>
    13: </ingredients>
    14: <process><P>Begin by sauteing the celery and onions in butter until soft.

```

```

15: Add oysters, oyster liquor, and cream. Heat until the oysters float.
16: Serve in warm bowls.</p>
17: <p><i>Yummy!</i></p></i>
18: </process>

```

## Exercice 2 (DTD from XML) :

Écrivez une DTD pour le document XML suivant :

```

1: <?xml version="1.0"?>
2: <shiporder orderid="889923">
3:   <orderperson>John Smith</orderperson>
4:   <shipto>
5:     <name>Ola Nordmann</name>
6:     <address>Langgt 23</address>
7:     <city>4000 Stavanger</city>
8:     <country>Norway</country>
9:   </shipto>
10:  <item>
11:    <title>Empire Burlesque</title>
12:    <note>Special Edition</note>
13:    <quantity>1</quantity>
14:    <price>10.90</price>
15:  </item>
16:  <item>
17:    <title>Hide your heart</title>
18:    <quantity>1</quantity>
19:    <price>9.90</price>
20:  </item>
21: </shiporder>

```

## Exercice 3 (XML from DTD) :

Écrivez un document XML basé sur la DTD suivante :

```

1: <?xml version="1.0"?>
2: <!DOCTYPE stock[
3:   <!ELEMENT stock (new-car | used-car)*>
4:   <!ELEMENT new-car (model, price)>
5:   <!ELEMENT used-car (model, price, mileage, condition?)>
6:   <!ELEMENT model (#PCDATA)>
7:   <!ELEMENT price (#PCDATA)>
8:   <!ELEMENT mileage (#PCDATA)>
9:   <!ELEMENT condition (#PCDATA)>
10: ]>

```

## Exercice 4 (XML & DTD) :

Envisager une application dans laquelle les résultats des matchs de football doivent être représentés en XML.

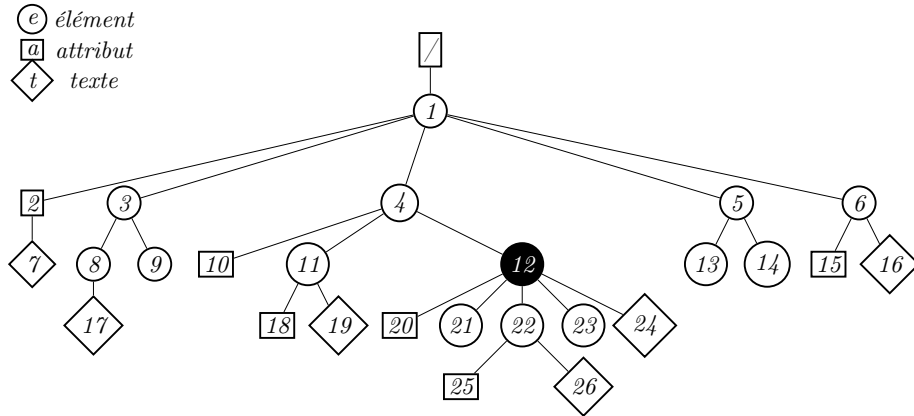
Pour chaque **jeu**, nous voulons être en mesure de représenter **les deux équipes** impliquées, l'équipe qui jouait **chez eux**, quels **joueurs** ont marqué des buts (dont certains peuvent avoir été **pénalités**) et le **moment** où chacun a été marqué, et quels joueurs ont reçu des **cartons jaunes** et **rouges**.

Écrivez un document XML et une DTD pour cette application. Vous pouvez utiliser des attributs.

Assurez-vous que votre document est bien écrit et valide à l'aide d'un programme qui valide votre document.

### Exercice 5 (XPath – axe) :

Regardez l'arbre XML ci-dessous. Notez les nœuds couverts par chaque axe, à partir de nœud 12.



- ancestor :
- ancestor-or-self :
- attribute :
- child :
- descendant :
- descendant-or-self :
- following :
- following-sibling :
- parent :
- preceding :
- preceding-sibling :
- self :

### Exercice 6 (XPath) :

Expliquez la signification des expressions suivantes :

- //COURS[INTITULE='XML']
- //COURS[INTITULE=XML]

Donnez une instance pour laquelle le résultat est identique.

### Exercice 7 (XPath) :

Expliquez la différence entre les deux expressions suivantes, et donnez un document pour laquelle on n'obtient pas le même résultat :

- //B[position() = 1]
- /descendant::B[position() = 1]

### Exercice 8 (XPath) :

Considérez la DTD suivante pour le document *Films.xml* décrivant des données cinématographiques :

```

1: <!DOCTYPE FILMS [
2: <!ELEMENT FILMS (FILM+, ARTISTE+)>
3: <!ELEMENT FILM (TITRE, GENRE, PAYS, MES, ROLES, RESUME?)>
4: <!ELEMENT TITRE (#PCDATA)>

```



```

5: <!ATTLIST FILM Annee CDATA #REQUIRED>
6: <!ELEMENT GENRE (#PCDATA)>
7: <!ELEMENT PAYS (#PCDATA)>
8: <!ELEMENT MES (#PCDATA)>
9: <!ATTLIST MES id_mes IDREF #IMPLIED>
10: <!ELEMENT ROLES (ROLE*)>
11: <!ELEMENT ROLE (PRENOM, NOM, INTITULE)>
12: <!ELEMENT PRENOM (#PCDATA)>
13: <!ELEMENT NOM (#PCDATA)>
14: <!ELEMENT INTITULE (#PCDATA)>
15: <!ELEMENT RESUME (#PCDATA)>
16: <!ELEMENT ARTISTE (ACTNOM, ACTPNOM, ANNEENAISS)>
17: <!ATTLIST ARTISTE id_art ID #REQUIRED>
18: <!ELEMENT ACTNOM (#PCDATA)>
19: <!ELEMENT ACTPNOM (#PCDATA)>
20: <!ELEMENT ANNEENAISS (#PCDATA)>
21: ]>

```

*Appliquez à ce document les expressions XPath pour rechercher les informations suivantes :*

1. La liste des titres de films.
2. Les titres des films parus en 1990.
3. Le résumé d'Alien.
4. Les titre des films avec Bruce Willis.
5. Les titres des films qui ont un résumé.
6. Les titres des films qui n'ont pas de résumé.
7. Les titres des films vieux de plus de trente ans.
8. Quel rôle joue Harvey Keitel dans Reservoir dogs ?
9. Quel est le dernier film du document ?
10. Quel est le titre du film qui précède immédiatement Shining (dans l'ordre du document) ?
11. Qui a mis en scène Vertigo ?
12. Les titres des films qui contiennent un « V » (utiliser la fonction contains())
13. Les noeuds qui ont exactement trois descendants (utiliser la fonction count()).
14. Les noeuds dont le nom contient la chaîne « TU » (fonction name())