DTD et XML Schémas
un exemple: VOTable

François Ochsenbein

CDS, Observatoire Astronomique

C’est quoi, XML ?

- Formalisation de SGML/HTML pour en faire un langage structuré (hiérarchique)
- Principe des `<balises>` (tags ou *éléments*) `</balises>` qui permettent une description associée aux données (les *métadonnées*) dans des documents.
- Les éléments peuvent contenir des attributs permettant certaines qualifications, par exemple

```
<livre> <titre> La formule magique </titre>
<prix devise=’Euro’>50</prix>
</livre>
```

Exemple (extrait VOTable)

```
<!DOCTYPE VOTABLE SYSTEM "http://us-vo.org/xml/VOTable.dtd">
 <VOTABLE version="1.0">
 <DESCRIPTION>VizieR Server  2002-04-24T17:18:02
 </DESCRIPTION>
 <!-- VOTable description at
   http://vizier.ustrasbg.fr/doc/VOTable/ -->
<DEFINITIONS>
<COOSYS ID="J2000" system="eq_FK5" equinox="J2000"/>
</DEFINITIONS>
<!-- Parameters used for the Query:
  -source=Veron RA=0 DEC=90 SR=2 VERB=1   -->
<INFO ID="Ref" name="-ref" value="VOTx9120" />
<INFO ID="Target" name="-c"
   value="000.000000+90.000000,rm=120." />
<RESOURCE ID="yCat_7224" name="VII/224" type="results">
```

Eléments -Attributs

- Elements sont entièrement contenus les uns dans les autres `<VOTABLE> … </VOTABLE>`
- Si une entité n'a pas de contenu, elle peut inclure le / comme pour `<INFO/>` ou `<BR/>`
- Les Attributs sont des « qualifications » des éléments, par exemple l'equinoxe dans

```
<COOSYS ID="J2000" system="eq_FK5" equinox="J2000"/>
```

```
<VOTABLE version="1.0">
<RESOURCE ID="yCat_7224" name="VII/224">
  <DESCRIPTION>Quasars and Active Galactic Nuclei (10th Ed.) (Veron+ 2001)</DESCRIPTION>
  <TABLE ID="VII_224_table1" name="VII/224/table1">
    <DESCRIPTION>Quasars (brighter than absolute magnitude -23)</DESCRIPTION>
    <FIELD name="_r" ucd="POS_ANG_DIST_GENERAL" datatype="float" width="7" precision="5"
unit="deg">
    <DESCRIPTION>Distance from center (00 00 00.0 +90 00 00) </DESCRIPTION>
    </FIELD>
    <FIELD name="_RAJ2000" ucd="POS_EQ_RA_MAIN" ref="J2000" datatype="float" width="8"
precision="4" unit="deg">
      <DESCRIPTION>Right ascension (FK5) Equinox=J2000.</DESCRIPTION>
    </FIELD>
    <FIELD name="_DEJ2000" ucd="POS_EQ_DEC_MAIN" ref="J2000" datatype="float" width="8"
precision="4" unit="deg">
      <DESCRIPTION>Declination (FK5) Equinox=J2000.</DESCRIPTION>
    </FIELD>
<DATA><TABLEDATA>
<TR><TD>4.59722</TD><TD>024.4050</TD><TD>+85.4028</TD></TR>
<TR><TD>4.80111</TD><TD>168.3900</TD><TD>+85.1989</TD></TR>
</TABLEDATA></DATA>
</TABLE>
</RESOURCE></VOTABLE>
```

Le DTD

- *Document Type Definition* permet de définir la structure d'un document: pour permettre sa vérification et sa *validation*

Ce qu'on trouve à `http://us-vo.org/xml/VOTable.dtd` :

```
<!ELEMENT VOTABLE (DESCRIPTION?, DEFINITIONS?, INFO*,
    RESOURCE*)>
<!ATTLIST VOTABLE
                ID ID #IMPLIED
  version CDATA #IMPLIED >
<!ELEMENT RESOURCE (DESCRIPTION?, INFO*, COOSYS*, PARAM*,
    LINK*, TABLE*, RESOURCE*)>
<!ATTLIST RESOURCE
      name CDATA #IMPLIED
      ID ID #IMPLIED >
```

Le DTD comme validateur

- Permet de vérifier que les *éléments* sont bien présents et bien dans l'ordre <VOTABLE> <RESOURCE> …

  avec précision 1fois, 0 ou 1 fois (?) 0 ou n fois (*), 1 ou n fois (+)
- Permet de vérifier la présence des *attributs* , et que le contenu des attributs soit conforme à une liste

…**mais ne permet pas de vérification détaillée** – par exemple qu'il y ait bien un nombre dans un prix

```
<prix devise="Euro">50</prix>
<prix devise="Euro">cinquante</prix>
```

Exemple (extrait VOTable)

```
<VOTABLE xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://us-vo.org/xml/VOTable.xsd">
 <VOTABLE version="1.0">
 <DESCRIPTION>VizieR Server  2002-04-24T17:18:02
 </DESCRIPTION>
 <!-- VOTable description at
   http://vizier.ustrasbg.fr/doc/VOTable/ -->
<DEFINITIONS>
<COOSYS ID="J2000" system="eq_FK5" equinox="J2000"/>
</DEFINITIONS>
<!-- Parameters used for the Query:
  -source=Veron RA=0 DEC=90 SR=2 VERB=1   -->
<INFO ID="Ref" name="-ref" value="VOTx9120" />
<INFO ID="Target" name="-c" value="000.000000+90.000000,rm=120."
   />
<RESOURCE ID="yCat_7224" name="VII/224" type="results">
```

XMLSchémas

- Recommandation W3C
- http://www.w3.org/TR/xmlschema-0
- Permet de décrire une structure de document XML d'un façon très précise **en XML** *(par ex. spécification des types de données autorisés, des intervalles de valeurs, de « patterns », etc…)*
- C'est une base de description utilisée dans d'autres standards en développement rapide (WSDL, UDDI, …)
- outils existants, par ex. XML Spy, Xerces,… pour générer/parser du XML, convertir DTDs…

Une déclaration XML-Schéma

```
<xs:element name="VOTABLE">
<xs:complexType>
<xs:sequence>
  <xs:element ref="DESCRIPTION" minOccurs="0"/>
  <xs:element ref="DEFINITIONS" minOccurs="0"/>
  <xs:element ref="INFO" minOccurs="0" maxOccurs="unbounded"/>
  <xs:element ref="RESOURCE" minOccurs="0" maxOccurs="unbounded" />
 </xs:sequence>
<xs:attribute name="ID" type="xs:ID"/>
<xs:attribute name="version">
 <xs:simpleType> <xs:restriction base="xs:NMTOKEN"> <xs:enumeration
     value="1.0"/> </xs:restriction></xs:simpleType>
</xs:attribute> </xs:complexType> </xs:element>
```

```
<!ELEMENT VOTABLE (DESCRIPTION?, DEFINITIONS?, INFO*, RESOURCE*)>
<!ATTLIST VOTABLE
                ID ID #IMPLIED
  version CDATA #IMPLIED >
```

XMLSchema: définition des éléments

Les éléments peuvent être

- **Simples**: ne contiennent que du texte

  Exemple: `<prix>50</prix>`

- **Complexes**: incluent des sous-éléments et/ou des attributs

  Exemples: `<prix devise="Euro">50</prix>`

```
<VOTABLE version="1.0">
 <RESOURCE name="Tycho-2"> …
 </RESOURCE>
</VOTABLE>
```

Elements Simples

- Définition utilisant des types de base

```
<prix>50.0</prix>
```

```
<xs:element name="prix" type="xs:decimal"/>
```

- Possibilités de rajouter des contraintes sur un type *(restriction)*

```
<xs:element name="prix">
  <xs:simpleType>
    <xs:restriction base="xs:decimal">
      <xs:fractionDigits value='2' fixed='true'>
    </xs:restriction></xs:simpleType>
</xs:element>
```

Elements simples

- Les restrictions peuvent comprendre:
  - Une liste de valeurs autorisées par **`enumeration`**

```
<xs:simpleType name='class'>
 <xs:restriction base='xs:NMTOKEN'>
  <xs:enumeration value='Star'/>
  <xs:enumeration value='Galaxy'/>
 </xs:restriction></xs:simpleType>
```

```
<class>
Star
</class>
```

  - des limites pour les nombres comme `minInclusive` `maxExclusive` ...
  - des expressions régulières *à la Perl* pour les chaînes de caractères *(équinoxe astronomique)*

```
<xs:pattern value='[JB][12][0-9]{3}(\.\d*)?'/>
```

Les Types de Base

- xs:string

Types dérivés:

xs:normalizedString

xs:token

xs:NMTOKEN

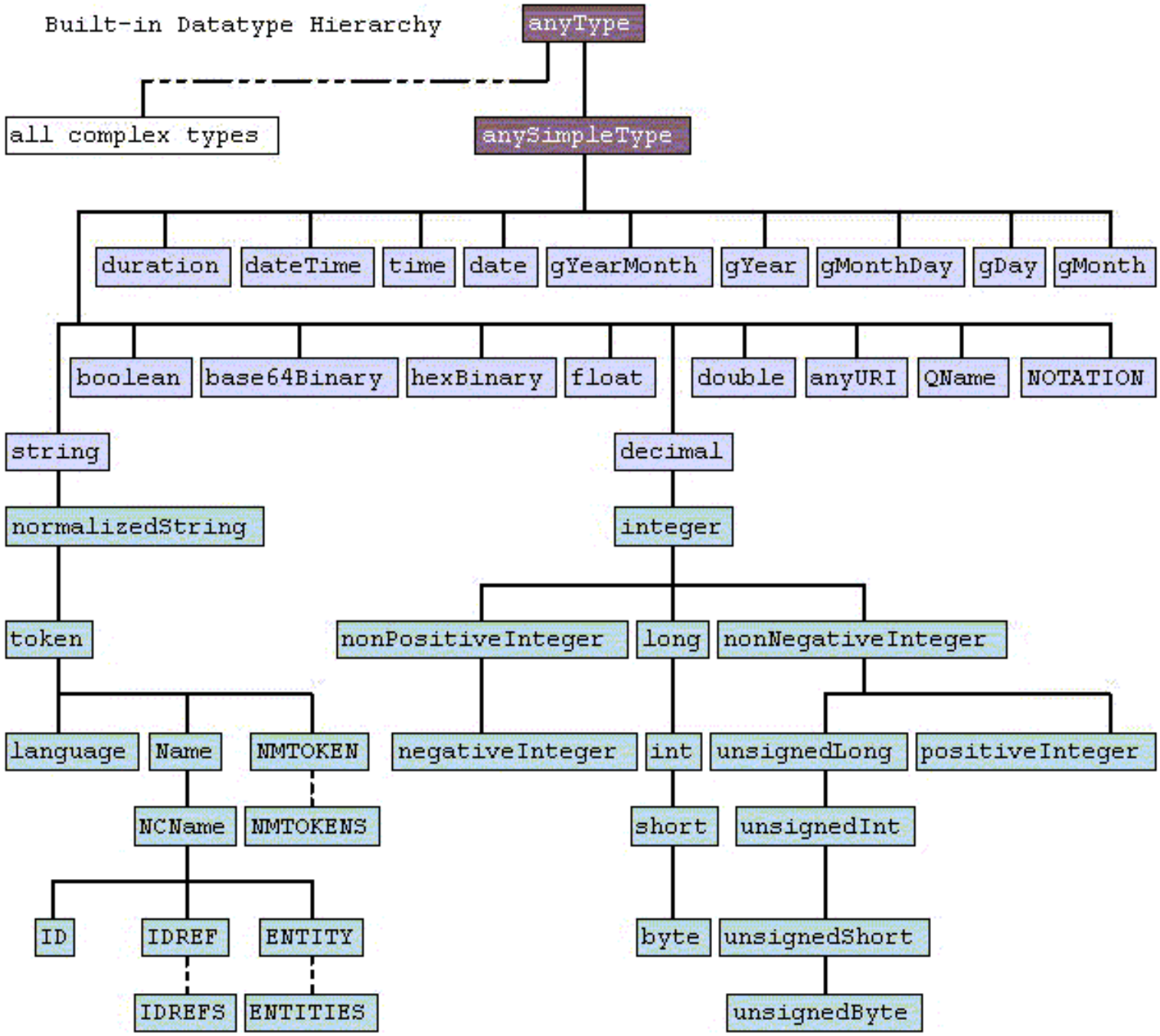
xs:ID

- xs:float
- xs:double
- xs:decimal
- xs:boolean
- xs:dateTime
- xs:duration

Types dérivés:

integer short…

nonNegativeInteger…


Built-in Datatype Hierarchy
anyType
all complex types
anySimpleType
duration
dateTime
time
date
gYearMonth
gYear
gMonthDay
gDay
gMonth
boolean
base64Binary
hexBinary
float
double
anyURI
QName
NOTATION
string
decimal
normalizedString
integer
token
nonPositiveInteger
long
nonNegativeInteger
language
Name
NMTOKEN
negativeInteger
int
unsignedLong
positiveInteger
NCName
NMTOKENS
short
unsignedInt
ID
IDREF
ENTITY
byte
unsignedShort
IDREFS
ENTITIES
unsignedByte

Les Types Complexes

- permettent de définir les hiérarchies d'éléments – mais sont aussi nécessaires pour déclarer des éléments avec attributs
- Exemple:

```
<xs:element name="VOTABLE"> <xs:complexType>
<xs:sequence>
  <xs:element ref="DESCRIPTION" minOccurs="0"/>
  <xs:element ref="DEFINITIONS" minOccurs="0"/>
  <xs:element ref="INFO" minOccurs="0" maxOccurs="unbounded"/>
  <xs:element ref="RESOURCE" minOccurs="0" maxOccurs="unbounded" />
 </xs:sequence>
</xs:complexType> </xs:element>
```

Séquences

- Imbrication des éléments peut être précisée par

  <xs:sequence> une suite dans l'ordre précisé

  <xs:choice> un des éléments proposés

  <xs:all> l'ordre des éléments est indifférent

- Contrôle de la répétition par minOccurs et maxOccurs qui ont la valeur 1 par défaut.

  ? minOccurs="0"

  * minOccurs="0" maxOccurs="unbounded"

  + maxOccurs="unbounded"

Les Attributs

Nécessitent une définition en **complexType**

```
<xs:element name='prix'>
 <xs:complexType>
  <xs:simpleContent>
   <xs:extension base:'xs:decimal'>
    <xs:attribute name='devise' default='Euro'>
     <xs:simpleType>
      <xs:restriction base='xs:NMTOKEN'>
       <xs:enumeration value='Euro'/>
        …
       </xs:restriction>
      </xs:simpleType>
     </xs:attribute>
    </xs:extension>
   </xs:simpleContent>
 </xs:complexType>
</xs:element>
```

Les définitions de nouveaux types

Déclaration

```
<xs:simpleType name='deviseForte'>
 <xs:restriction base='NMTOKEN'>
  <xs:enumeration value='Euro'/>
  <xs:enumeration value='USD'/>
  <xs:enumeration value='GBPound'/>
 </xs:restriction>
</xs:simpleType>
```

Utilisation

```
<xs:element name='prix'>
<xs:complexType><xs:simpleContent>
  <xs:extension base:'xs:decimal'>
   <xs:attribute name='devise'
       type='deviseForte'>
  </xs:extension></xs:simpleContent>
 </xs:complexType>
</xs:element>
```

Définitions d'éléments

Déclaration

```
<xs:complexType name='VALUE'>
 <xs:simpleContent>
  <xs:extension base:'xs:decimal'>
   <xs:attribute name='unit'
       type=xs:NMTOKEN'/>
  </xs:extension>
 </xs:simpleContent>
</xs:complexType>
```

Utilisation

```
<MEASURES>
<VALUE unit='km/s'>
 52.3</VALUE>
<VALUE unit='K'>
 5040</VALUE>
</MEASURES>
```

```
<xs:element name='MEASURES'>
 <xs:complexType>
  <xs:sequence>
   <xs:element type='VALUE'
      maxOccurs='unbounded'/>
  </xs:sequence>
 </xs:complexType>
 </xs:element>
```

```
<xs:element name="PARAM">
   <xs:complexType>
     <xs:sequence>
       <xs:element ref="DESCRIPTION" minOccurs="0"/>
       <xs:element ref="VALUES" minOccurs="0"/>
       <xs:element ref="LINK" minOccurs="0" maxOccurs="unbounded"/>
     </xs:sequence>
     <xs:attribute name="ID" type="xs:ID"/>
     <xs:attribute name="unit" type="xs:token"/>
     <xs:attribute name="datatype" type="dataType"/>
     <xs:attribute name="precision" type="precType"/>
     <xs:attribute name="width" type="xs:positiveInteger"/>
     <xs:attribute name="ref" type="xs:IDREF"/>
     <xs:attribute name="name" type="xs:token" use="required"/>
     <xs:attribute name="ucd" type="xs:token"/>
     <xs:attribute name="value" type="xs:string"/>
     <xs:attribute name="arraysize" type="arrayDEF"/>
   </xs:complexType>
 </xs:element>
```

Modèles de définitions

- « poupées russes » éléments imbriqués *anonymes* – peut amener à des répétitions.
- « salami »: déclarations des éléments *simples*, utilisés par `ref=` dans les structures complexes *(ce qui reflète le mieux une description DTD)*
- « store vénitien »: définition des types utilisés avec `type=`

Quelques réflexions…

- XMLSchema permet de décrire précisément des structures pour l'échange de données, mais quelle lourdeur !
- Il existe transformateurs de DTD en XML-Schéma (par exemple *XMLSpy*) – le code généré est encore plus lourd!