Bases de données semi-structurées

Master 2 Humanités numériques

2024-2025

Jérôme Darmont

https://eric.univ-lyon2.fr/jdarmont/

Actualité du cours

https://eric.univ-lyon2.fr/jdarmont/?page_id=3135

https://eric.univ-lyon2.fr/jdarmont/?feed=rss2

https://social.sciences.re/@darmont #hnbdss

Plan

- Introduction : Données semi-structurées
- Langage XML
- Langage XQuery

Données structurées

- Données organisées en entités
- Entités similaires : forment des groupes (classes)
- Entités du même groupe : même description (attributs)
- Pour toutes les entités d'un groupe :
  - Chaque attribut a le même type
  - Chaque valeur d'attribut a la même taille
  - Tous les attributs sont présents
  - Les attributs sont toujours dans le même ordre

- Données structurés : décrites par un schéma
  - Généralement stockées dans des bases de données

Données non structurées

- Données de tous types
- Données qui ne suivent aucun schéma ni séquence prédéterminé
- Données qui ne suivent aucune règle
- Données qui ne sont pas prévisibles

- Exemples de données non-structurées :
  - Textes
  - Images
  - Vidéos
  - Sons

Données semi-structurées (1/2)

Bases de données

Documents

Langages de requête

Moteurs de recherche

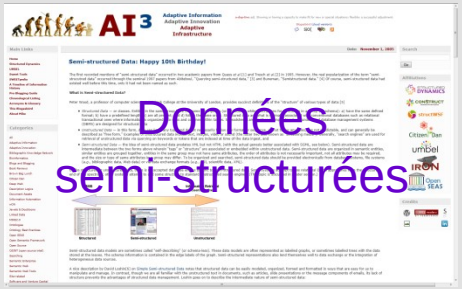
Années 1990

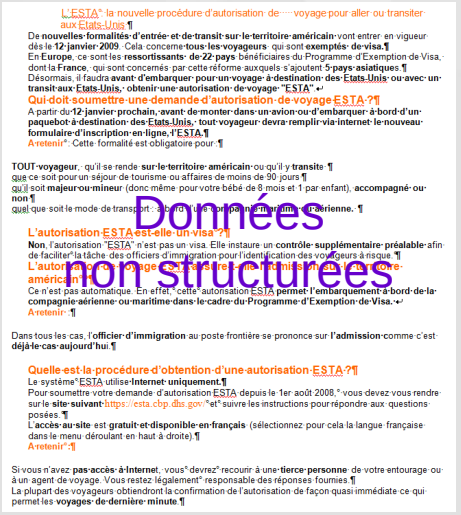
Données structurées

Données semi-structurées

Données non structurées

N° employé	Date embauche	Nom	Prénom	Adresse	Ville	Provinc	Code post	Date naissan
0090	12-janv-89	Charest	Eric	1075 rue Duluth	Montréal	QC	H4T 2L9	06-sept-49
0101	20-févr-89	Reys	Kimberly	899 ch. lakeshore	Donval	QC	H9P 3T2	15-déc-53
0150	05-mars-89	Langlois	Claude	308 rue Ste-Pierre	Montréal	QC	H2P 1A6	03-mars-68
0151	14-oct-89	Bédard	Julie	440 rue Talbot	Laval	QC	H7P 3G8	14-juil-45
0160	07-mars-89	Fortin	Jean-Paul	4380 rue Raccot	Montréal-Nord	QC	H2Y 2P5	25-juil-59
0230	19-mars-89	Labrecque	Monique	892 ave. Victoria	Longueuil	QC	J3P 2T2	02-févr-37
0234	12-avr-89	Therrien	Jacques	320 rue Verchères	Greenfield Park	QC	J4V 2V6	02-févr-71
0367	21-avr-89	Proulx	Marcel	8085 boul. St-Laurent	Montréal	QC	H3Y 2W3	06-mars-56
0380	12-mai-89	Labranche	Andrée	99 rue du Collège	Ville St-Laurent	QC	H4P 2M6	05-juil-64
0434	04-juin-89	Connell	Mark	1213 rue de Gaulle	St-Hubert	QC	J4T 2L9	28-févr-59
0600	10-juin-89	Reynolds	Alain	868 rue Ouellette	Outremont	QC	H3V 2V2	25-juil-67
0650	15-juin-89	Simard	Martin	2575 rue Victoria	St-Lambert	QC	J4L 6P7	03-avr-74
0728	15-juil-89	Bédard	Sébastien	3529 rue Rouvière	Ste-Thérèse	QC	J7P 3V5	14-janv-53
0839	14-août-89	Lebrun	Josée	694 de Montarville	St-Bruno	QC	J4L 3Y4	12-sept-71
1142	10-oct-89	Faucher	Martin	45 rue Handel	Candiac	QC	J2Y 2M7	31-oct-54
1151	14-oct-89	Bédard	Julie	621 rue Cannes	Laval	QC	H7P 3G7	02-juil-48
1241	07-janv-90	Simard	Martin	45 rue Handel	Candiac	QC	J3V 5P9	25-nov-50
1260	01-févr-90	Morissette	Jean-Guy	2358 rue Roquebrune	Greenfield Park	QC	H3W 2X3	06-févr-58
1341	04-mars-90	Verreault	Serge	1067 rue Lapierre	Lasalle	QC	J4P 1M2	13-mai-65
1460	05-nov-90	Quinn	Francis	4878 Côte-des-Neiges	Montréal	QC	H3Y 2P2	28-août-65
1473	07-nov-90	Vaudrin	Jean-Claude	2689 rue Fleury	Montréal	QC	H4M 1C5	12-déc-52
1481	01-déc-90	Wasty	Maryse	383 de LaSalle	Boucherville	QC	J4Z 1T7	07-sept-68
1543	09-janv-91	Raymond	Louis	4789 rue Sherbrooke est	Montréal	QC	H2T 2V6	30-mai-66
1560	08-juil-91	Bandi	Ahmed	1432 rue Principale	Granby	QC	H7P 7P8	03-juin-55
1571	14-août-91	Marcos	Hélène	1637 boul. Wilfrid-Hamel	Québec	QC	G0X 2X0	17-oct-62
2300	16-août-91	Sicotte	Hélène	1546 rue Bélanger	Montréal	QC	J4P 1X3	26-sept-53
2312	04-nov-91	Lebrun	Carmen	1341 rue Nobert	Montréal	QC	H3L 5M3	03-août-70
2315	04-nov-91	Couillard	Sophie	2689 boul. Décarie	Ville St-Laurent	QC	H4Y 2Z2	28-juil-59
2320	04-nov-91	Hamel	Nathalie	365 ave. Hutchison	Outremont	QC	H2V 2T4	15-mars-51
2321	01-déc-91	Milot	Bernard	8967 rue Viger	Montréal	QC	H3W 2M5	10-mai-73
2322	01-déc-91	Arel	Gilbert	436 rue Dawson	Montréal	QC	H3L 6L9	19-oct-67
2340	01-déc-91	Ouellet	Patricia	7228 boul. St-Joseph	Montréal	QC	H2G 3P6	13-avr-51
2341	05-janv-92	Archambault	Manon	673 Terr. Beaupré	Montréal	QC	H4G 2T1	31-déc-62
3389	04-févr-92	Brodeur	Guy	2575 Tisserand	Brossard	QC	J4P 2V3	26-juin-60
3490	04-mars-92	Lagacé	Mireille	592 St-André	Montréal	QC	H2T 5X2	29-févr-73
3500	04-mars-92	Paquette	Pierre	3687 Christophe-Colomb	Montréal	QC	H3V 5M1	09-nov-72
3501	04-sept-92	Trottier	Laurent	1955 rue Villeneuve	Longueuil	QC	J3P 6Y3	11-janv-72
3550	15-nov-92	Tremblay	Alexandra	524 ave. Dumont	Boucherville	QC	J4Z 8M2	14-févr-69
3554	15-nov-92	Voisard	François	2602 rue Perron	Montréal	QC	H4T 3M9	30-mars-66
3560	21-nov-92	Weber	Hans	52 rue Ste-Marie	Verchères	QC	J5G 4X2	20-sept-72
3561	21-nov-92	Gagnon	Eric	2235 rue Bréheuf	Montréal	QC	H3P 2J5	06-août-46
3580	09-déc-92	Demers	Marie-Claire	1275 rue Van Horne	Outremont	QC	H2V 6P5	05-nov-58
3800	08-déc-92	Robitaille	Céline	7629 Édouard-Montpetit	Montréal	QC	H3V 0L3	29-juin-74





L'ESTA* la nouvelle procédure d'autorisation de voyage pour aller ou transiter aux Etats-Unis

De nouvelles formalités d'entrée et de transit sur le territoire américain vont entrer en vigueur dès le 12 janvier 2009. Cela concerne tous les voyageurs qui sont exemptés de visa.

En Europe, ce sont les ressortissants de 22 pays bénéficiaires du Programme d'Exemption de Visa, dont la France, qui sont concernés par cette réforme auxquels s'ajoutent 5 pays asiatiques.

Désormais, il faudra avant d'embarquer pour un voyage à destination des Etats-Unis ou avec un transit aux Etats-Unis, obtenir une autorisation de voyage "ESTA".

Qui doit soumettre une demande d'autorisation de voyage ESTA ?

A partir du 12 janvier prochain, avant de monter dans un avion ou d'embarquer à bord d'un paquebot à destination des Etats-Unis, tout voyageur devra remplir via internet le nouveau formulaire d'inscription en ligne, l'ESTA.

A retenir* : Cette formalité est obligatoire pour :

TOUT voyageur, qu'il se rende sur le territoire américain ou qu'il y transite ; que ce soit pour un séjour de tourisme ou affaires de moins de 90 jours ; qu'il soit majeur ou mineur (donc même pour votre bébé de 8 mois et 1 par enfant), accompagné ou non ; quel que soit le mode de transport : aérien, maritime ou terrestre.

L'autorisation ESTA est elle un visa ?

Non, l'autorisation "ESTA" n'est pas un visa. Elle instaure un contrôle supplémentaire préalable afin de faciliter la tâche des officiers d'immigration pour l'identification des voyageurs à risque.

L'autorisation "ESTA" ne remplace pas le visa de non immigrant délivré par les autorités américaines.

Ce n'est pas automatique. En effet, cette autorisation ESTA permet l'embarquement à bord de la compagnie aérienne ou maritime dans le cadre du Programme d'Exemption de Visa.

A retenir :

Dans tous les cas, l'officier d'immigration au poste frontière se prononce sur l'admission comme c'est déjà le cas aujourd'hui.

Quelle est la procédure d'obtention d'une autorisation ESTA ?

Le système ESTA utilise Internet uniquement.

Pour soumettre votre demande d'autorisation ESTA depuis le 1er août 2008, vous devez vous rendre sur le site suivant : https://esta.cbp.dhs.gov/ et suivre les instructions pour répondre aux questions posées.

L'accès au site est gratuit et disponible en français (sélectionnez pour cela la langue française dans le menu déroulant en haut à droite).

A retenir :

Si vous n'avez pas accès à Internet, vous devrez recourir à une tierce personne de votre entourage ou à un agent de voyage. Vous restez légalement responsable des réponses fournies.

La plupart des voyageurs obtiendront la confirmation de l'autorisation de façon quasi immédiate ce qui permet les voyages de dernière minute.

Données semi-structurées (2/2)

- Données organisées en entités sémantiques
- Entités similaires : groupes
- Entités du même groupe : peuvent ne pas avoir les mêmes attributs
- Pour toutes les entités d'un groupe :
  - Un même attribut peut avoir des types différents
  - Une même valeur d'attribut peut avoir des tailles différentes
  - Des attributs peuvent être manquants ou dupliqués
  - L'ordre des attributs n'est pas nécessairement important

Structurées

Non-structurées

- Données semi-structurées : autodescriptives
  - Pages web, documents XML, courriels…

Exemple de données semi-structurées

- Nom Jérôme Darmont
- Courriel jerome.darmont@univ-lyon2.fr
  jerome.darmont@msh-lse.fr

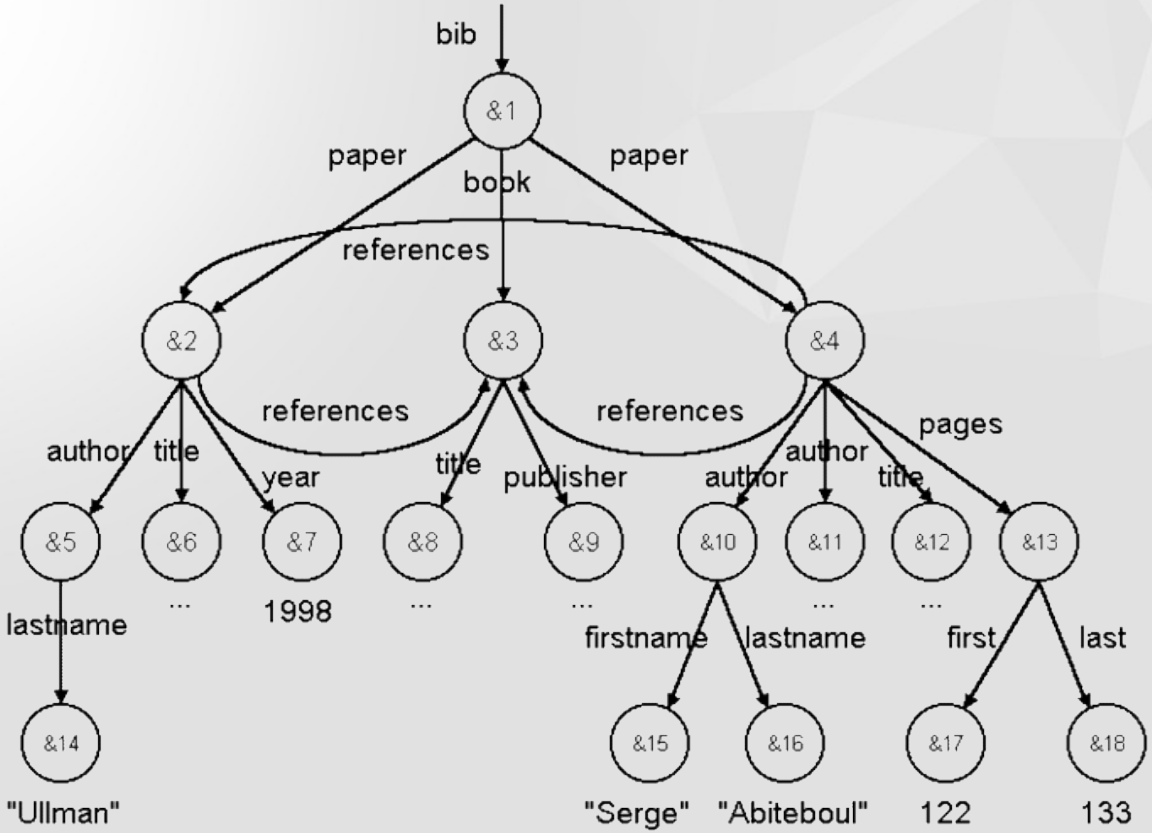
- Courriel sabine.loudcher@univ-lyon2.fr
- Nom
  - Prénom Loudcher
  - Nom de famille Sabine

- Nom Julien Velcin
- Affiliation Université Lyon 2

Modèle de données semi-structuré

- Avantages
  - Peut représenter des informations issues de sources de données qui ne peuvent pas êtres contraintes par un schéma
  - Format flexible pour l'interopérabilité
  - Permet de voir des données structurées comme semi-structurées (Web)
  - Schéma facilement évolutif
- Inconvénients
  - Performance des requêtes sur données à grande échelle
- Représentations
  - Electronic Data Interchange (EDI) : domaine financier
  - Object Exchange Model (OEM) : modèle basé sur les graphes
  - SGML, HTML et XML
  - JSON, YAML…

Exemple de graphe OEM



Gestion de données semi-structurées

- Modélisation des données semi-structurées
  - Graphes (OEM) Modèle conceptuel
  - DTD, XML-Schema Modèle logique
  - XML Modèle physique
- Requêtage des données semi-structurées
  - XPath
  - XQuery
- Stockage des données semi-structurées
  - Fichiers plats
  - Bases de données relationnelles, relationnelles-objets ou natives XML

Références

- Peter Wood, Birkbeck University of London
  Semi-Structured Data
  http://www.dcs.bbk.ac.uk/~ptw/

- Mike Bergman, Structured Dynamics LLC
  Semi-structured Data: Happy 10th Birthday!
  http://www.mkbergman.com/153/semi-structured-data-happy-10th-birthday/

Plan

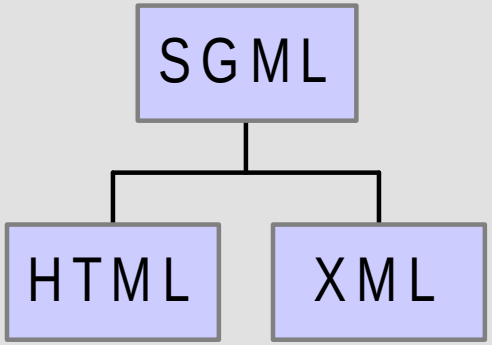
- ✔ Introduction
- Langage XML
- Langage XQuery

- Éléments
- Attributs
- Documents XML bien formés, documents valides
- DTD (*Document Type Definition*)
- XML Schema

Le langage XML

- XML : Extensible Markup Language
  - Format de structuration de données et de documents Internet issu de SGML
  - Définition, gestion, création, transmission et partage de documents

- XML est un standard du W3C
  - 1996 : Brouillon
  - 1997 : XML 1.0
  - 2004 : XML 1.1



Exemple de document XML

```
<?xml version="1.1"  encoding="utf-8" ?> <!-- Prologue (obligatoire) -->
<annuaireProfs> <!-- Élément racine -->
    <!– Sous-éléments  -->
    <prof>
        <nom>Jérôme Darmont</nom>
        <courriel>jerome.darmont@univ-lyon2.fr</courriel>
        <cours>Bases de données semi-structurées</cours>
        <cours>Programmation web backend</cours>
    </prof>
    <prof>
        <nom>Julien Velcin</nom>
        <courriel>julien.velcin@univ-lyon2.fr</courriel>
        <cours>Programmation orientée objet</cours>
    </prof>
    <!-- Etc. -->
</annuaireProfs> <!-- Balise de fin -->
```

Ensemble d'éléments imbriqués matérialisés par des balises

Règles d'écriture d'un document XML (1/2)

- Un document XML a un et un seul élément racine.
- Les éléments doivent être correctement emboîtés (les balises ouvrantes et fermantes ne doivent pas se chevaucher).
- Tout élément doit avoir une balise ouvrante et une balise fermante.
- Le nom d'un élément doit être identique dans la balise ouvrante et la balise fermante.
- Les noms d'éléments sont sensibles à la casse. Ils doivent commencer par une lettre ou par _ suivi(e) de lettres, de chiffres, de . , de – ou de _.

Règles d'écriture d'un document XML (2/2)

- Les noms d'éléments commençant par XML (dans toutes combinaisons de minuscules et majuscules) sont réservés à des fins de standardisation.
- Un document XML respectant ces règles est dit bien formé.
- Un document XML doit être bien formé !
- Un document XML peut de plus être valide s'il se conforme à la structure définie dans une DTD ou un Schéma XML.

Document Type Definition XML Schema

Schémas XML

DTD	XML Schema
Syntaxe spécifique (non XML)	Exprimé dans un document XML
Typage faible	Typage fort
Modélisation partielle impossible	Modélisation partielle possible
Interprétable par un·e utilisateur·trice humain·e	Conçu pour des traitements automatiques

Éléments XML (1/2)

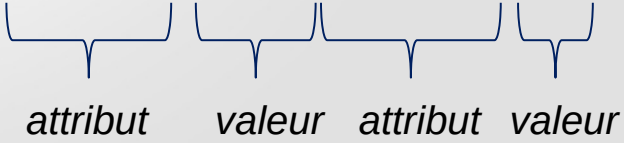
- Caractères non-autorisés : < &]]>
- Eléments emboîtés : profondeur non limitée
  - Ex.
```
<annuaireProfs>
    <prof>
        <nom>
            <nom_famille>Zighed</nom_famille>
            <prenom>Abdelkader</prenom>
            <prenom>Djamel</prenom>
        </nom>
    </prof>
</annuaireProfs>
```

Éléments XML (2/2)

- Section CDATA : Bloc de texte libre dans lequel seule la chaîne]]> est interdite
  - Ex.
```
<nom>
<![CDATA[<Darmont> & <Loudcher>]]>
</nom>
```
- Élément vide : sans contenu
  - Ex. <courriel></courriel>
  - Formulation équivalente <courriel />

Attributs d'éléments XML

- Attributs : données associées à un élément, complémentaires du contenu
- Définition : couple nom/valeur dans la balise ouvrante de l'élément
  - Ex. <bureau campus="PdA" batiment="K">063</bureau>



- Les attributs sont possibles dans les éléments vides.
  - Ex. <image source="ma-bobine.png" />

Contenu d'élément XML *vs.* attributs (1/2)

- Que choisir ?
  - <prof>
    <nom>Darmont</nom>
    </prof>
  - <prof nom="Darmont" />
- 4 principes pour décider
  - D'après Uche Ogbuji, Fourthought, Inc.

Contenu d'élément XML *vs.* attributs (2/2)

Principe	Élément	Attribut
Contenu principal	Information essentielle	Information périphérique
Information structurée	Information hiérarchisée	Information atomique
Lisibilité	Utilisateur.trice humain.e	Traitement automatique
Relation élément/attribut	Information précisée par une autre	

Exemple de relation élément/attribut

```
<stock>
    <produit quantité="1500">
        <nom>Ordinateur</nom>
    </produit>
    <produit quantité="500">
        <nom>Imprimante</nom>
    </produit>
</stock>
```

Documents XML valides

- Document valide
  - Bien formé
  - DTD (*Document Type Definition*) qui définit sa structure
  - L'élément racine se conforme à la structure définie dans la DTD
- DTD : prototype de document
  - Permet de vérifier automatiquement si un document XML est valide
  - Renforce l'uniformité d'un groupe de documents XML similaires

Spécification de DTD (1/6)

- DTD interne : incluse dans le document XML
  - Après le prologue et avant l'élément racine
  - <!DOCTYPE nom_élément_racine
    [
      <-- insérer la DTD ici -->
    ]>

- DTD externe : dans un fichier séparé
  - Préférable car la même DTD peut être utilisée pour plusieurs documents
  - Spécification dans le prologue du document
    <!DOCTYPE nom_élément_racine SYSTEM "URI">
  - Ex. <!DOCTYPE annuaireProfs SYSTEM "profs.dtd">

Spécification de DTD (2/6)

- Déclaration d'un type d'élément : <!ELEMENT nom contenu>
  - Ex. <!ELEMENT nom (#PCDATA)>

- Types de contenus possibles
  - Données textuelles : (#PCDATA)
  - Élément vide : EMPTY
  - Tout contenu légal : ANY
  - Séquence ordonnée : (élément$_1$, élément$_2$, …, élément$_n$)
    - Ex. <!ELEMENT prof (nom, courriel, cours)>
  - Choix : (élément$_1$ | élément$_2$ | … | élément$_n$)
    - Ex. <!ELEMENT uri (http | ftp | mailto | telnet)>
  - Note : les sous-éléments doivent être définis à leur tour

Spécification de DTD (3/6)

- Multiplicité des éléments :
  - 0 ou 1 occurrence : ?
  - 1 à n occurrences : +
  - 0 à n occurrences : *

- Exemples
  1. Élément montagne avec un ou plusieurs noms, une hauteur optionnelle et un pays obligatoire
     <!ELEMENT montagne (nom+, hauteur?, pays)>

  2. Élément montagne avec plusieurs occurrences des sous-éléments
     <!ELEMENT montagne (nom, hauteur, pays)*>

Spécification de DTD (4/6)

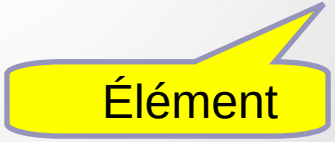
- Emboîtement de sous-éléments :
  - Ex. <!ELEMENT montagne (nom, hauteur, (pays, région, département))>

- Exemple global mais nonobstant simple

```
<!ELEMENT annuaireProfs (prof)*>
<!ELEMENT prof (nom, (courriel | tel), cours (titre, heures)+)>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT courriel (#PCDATA)>
<!ELEMENT tel (#PCDATA)>
<!ELEMENT titre (#PCDATA)>
<!ELEMENT heures (#PCDATA>
```

Spécification de DTD (5/6)

- Déclaration de liste d'attributs : <!ATTLIST nom_élt définitions_att>
  - Ex. <!ATTLIST bureau
    campus CDATA "Porte des Alpes"
    bâtiment CDATA #REQUIRED >



- Définition des attributs
  - Nom de l'attribut
  - Type d'attribut
  - Valeur par défaut (si #REQUIRED, pas de valeur par défaut)

Spécification de DTD (6/6)

- Types d'attributs
  - Chaîne de caractères : CDATA
  - Token
    - ID : L'attribut doit avoir une valeur unique pour chaque élément.
    - IDREF/IDREFS : La valeur de l'attribut référence la ou les valeurs de zéro ou plusieurs autre(s) attribut(s) de type ID. Les valeurs multiples sont séparées par des espaces.
  - Énumération : liste de valeurs possibles
    - Ex. <!ATTLIST bureau campus (Quai | Bron) "Bron">

Entités

- Entité : constante définie dans une DTD qui peut être référencée partout dans un document XML qui se conforme à cette DTD

- Définition dans la DTD : <!ENTITY nom valeur>
  - Ex. <!ENTITY dom-ull2 "univ-lyon2.fr">

- Référence dans le document XML : &nom;
  - Ex. <courriel>jerome.darmont@&dom-ull2;</courriel>
    <siteweb>https://eric.&dom-ull2/jdarmont/</siteweb>

Entités

- Entité : constante définie dans une DTD qui peut être référencée partout dans un document XML qui se conforme à cette DTD

- Définition dans la DTD : <!ENTITY nom valeur>
  - Ex. <!ENTITY dom-ull2 "univ-lyon2.fr">

- Référence dans le document XML : &nom;
  - Ex. <courriel>jerome.darmont@&dom-ull2;</courriel>
    <siteweb>https://eric.&dom-ull2/jdarmont/</siteweb>

XML Schema

- Limites des DTD
  - Syntaxe spécifique (non XML)
  - Typage faible
  - Pas de modélisation partielle
  - Mais lisible par des êtres humains !

- Caractéristiques de XML Schema
  - C'est un document XML.
  - Typage fort
  - Modélisation partielle possible
  - Plutôt utilisé par des machines…

Document XML Schema

- Structure du schéma dans un fichier séparé

```
<?xml version="1.0" encoding="utf-8" ?> <!-- mon_schema.xsd -->

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
        <!-- Définition des éléments -->
</xsd:schema>
```

- Référence au schéma dans un document XML

```
<racine    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="mon_schema.xsd">
```

Spécification des éléments

- Type simple

```
<xsd:element name="nom" type="xsd:string" />
```

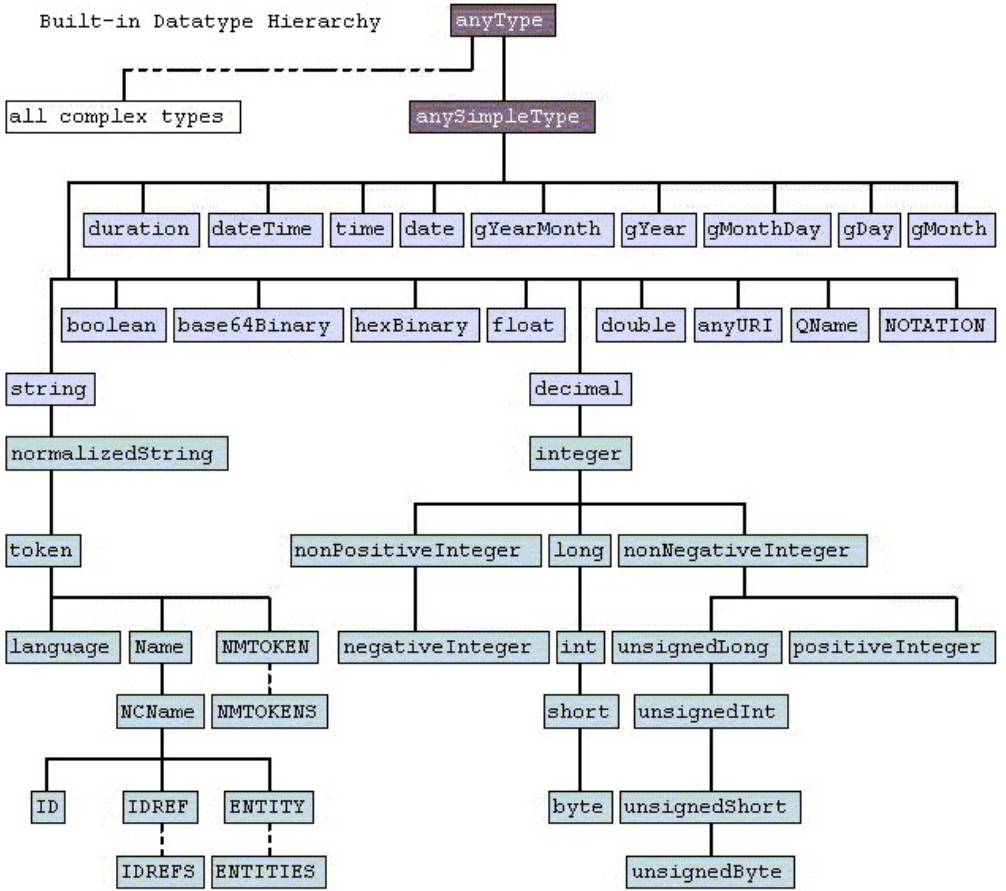
- Nombre d'occurrences

```
<xsd:element    name="cours" type="xsd:string"
                minOccurs="1" maxOccurs="unbounded" />
<!-- minOccurs et maxOccurs sont des entiers tels que maxOccurs ≥ minOccurs -->
```

- Type complexe

```
<xsd:element name="prof">
        <xsd:complexType>
                <!-- Spécification du type complexe (à suivre) -->
        </xsd:complexType>
</xsd:element>
```

Types de données simples



http://www.tbray.org/ongoing/When/200x/2009/04/29/-big/XSD-data-types.jpg

Types complexes (1/2)

- Séquence

```
<xsd:sequence>
        <xsd:element name="nom" type="xsd:string" />
        <xsd:element name="cours" type="xsd:string" />
        <xsd:element name="courriel" type="xsd:string" />
</xsd:sequence>
```

- Choix

```
<xsd:choice>
        <xsd:element name="http" type="xsd:string" />
        <xsd:element name="ftp" type="xsd:string" />
        <xsd:element name="mailto" type="xsd:string" />
        <xsd:element name="telnet" type="xsd:string" />
</xsd:choice>
```

Types complexes (2/2)

- Tout

```
<-- Chaque élément apparaît au moins une fois, mais dans n'importe quel ordre -->
<xsd:all>
        <xsd:element name="addresse" type="xsd:string" />
        <xsd:element name="courriel" type="xsd:string" />
        <xsd:element name="tel" type="xsd:string" />
</xsd:all>
```

- Référence à un élement complexe

```
<xsd:element ref="prof" />
 <-- ... -->
<xsd:element name="prof">
        <-- Définition de type complexe -->
</xsd:element>
```

Spécification des attributs

- Définition

```
<xsd:attribute name="miseajour" type="xsd:date" default="2023-08-22" />
<xsd:attribute name="montant" type="xsd:integer" use="required" />
```

- Restriction de type

```
<xsd:attribute name="bureau" use="required">
        <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="K061"/>
                        <xsd:enumeration value="K062"/>
                        <xsd:enumeration value="K063"/>
                </xsd:restriction>
        </xsd:simpleType>
</xsd:attribute>
```

Déclaration des attributs (1/2)

- Dans des éléments de type simple

```
<xsd:element name="post-blog">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="date-envoi" type="xsd:date" />
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
```

Déclaration des attributs (2/2)

- Dans des éléments de type complexe

```
<xsd:element name="prof">
  <xsd:complexType>
    <xsd:sequence>
      <!-- Définition de sous-éléments -->
    </xsd:sequence>
    <xsd:attribute name="bureau" type="xsd:string" />
  </xsd:complexType>
</xsd:element>
```

- Par référence

```
<!-- Comme ci-dessus, mais à définir plus tard -->
 <xsd:attribute ref="bureau" use="required" />
```

Exemple de XML Schema (1/2)

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

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

  <xsd:element name="annuaireProfs">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref=“prof" minOccurs=“0"
                                    maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
```

Exemple de XML Schema (2/2)

```
<xsd:element name="prof">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="nom" type="xsd:string" />
      <xsd:element name="cours" type="xsd:string"
                   minOccurs="1" maxOccurs="unbounded" />
      <xsd:element name="courriel" type="xsd:string" />
    </xsd:sequence>
    <xsd:attribute name="bureau" type="xsd:string" />
  </xsd:complexType>
</xsd:element>

</xsd:schema>
```

Plan

- ✔ Introduction
- ✔ Langage XML
- Langage XQuery
  - Expressions XPath
  - Requêtes FLWOR
  - Requêtes complexes

Langage XQuery

- Langage de requêtes pour données XML
- Similarités avec SQL
- Conçu par le W3C
- Basé sur des expressions XPath (mêmes modèle de données, fonctions, opérateurs)
- Versions
  - 2007 : XQuery 1.0 ⊃ XPath 2.0
  - 2017 : XQuery 3.1 ⊃ XPath 3.1
- Standardisation en cours (standard de fait)
- Soutenu par les éditeurs de SGBD (Oracle, Microsoft, IBM…)

Document XML exemple (1/2)

```
<?xml version="1.1" encoding="utf-8" ?> <!-- le fichier s'appelle films.xml -->

<catVOD>

    <film sigJeune="-12">
        <titre>Blade runner</titre>
        <realisateur>Ridley Scott</realisateur>
        <annee>1982</annee>
        <langue>Anglais</langue>
        <prix>4.79</prix>
    </film>

    <film>
        <titre>La grande vadrouille</titre>
        <realisateur>Gérard Oury</realisateur>
        <annee>1966</annee>
        <duree>122</duree>
        <langue>Français</langue>
        <prix>9.82</prix>
    </film>                                    <!-- (…) -->
```

Document XML exemple (2/2)


```
<film sigJeune="-10">
        <titre>Le fabuleux destin d'Amélie Poulain</titre>
        <realisateur>Jean-Pierre Jeunet</realisateur>
        <annee>2001</annee>
        <duree>120</duree>
        <langue>Français</langue>
        <prix>4.99</prix>
</film>

<film sigJeune="-12">
        <titre>The big Lebowski</titre>
        <realisateur>Ethan Coen</realisateur>
        <realisateur>Joel Coen</realisateur>
        <annee>1997</annee>
        <duree>112</duree>
        <langue>Français</langue>
        <langue>Anglais</langue>
        <prix>9.82</prix>
</film>

</catVOD>
```

Expressions de chemins (1/3)

- Document XML entier
  doc("films.xml")/catVOD
  Résultat
  Tout le document

- Un élément donné
  doc("films.xml")/catVOD/film
  doc("films.xml")/catVOD/film/titre
  Résultat
  <titre>Blade runner</titre>
  <titre>La grande vadrouille</titre>
  <titre>Le fabuleux destin d'Amélie Poulain</titre>
  <titre>The big Lebowski</titre>

Expressions de chemins (2/3)

- Un attribut donné

  doc("films.xml")/catVOD/film/data(@sigJeune)

  Résultat

  -12 -10 -12

- Un élément donné quel que soit son niveau hiérarchique

  doc("films.xml")/catVOD//titre

  //titre

  Résultat

  <titre>Blade runner</titre>
  <titre>La grande vadrouille</titre>
  <titre>Le fabuleux destin d'Amélie Poulain</titre>
  <titre>The big Lebowski</titre>

Expressions de chemins (3/3)

- Tous les sous-éléments d'un élément

  //film/*

  Résultat

  <titre>Blade runner</titre>
  <realisateur>Ridley Scott</realisateur>
  <annee>1982</annee>
  <duree>117</duree>
  <langue>English</langue>
  <prix>4.79</prix>
  <titre>La grande vadrouille</titre>
  <realisateur>Gérard Oury</realisateur>
  <annee>1966</annee>
  <duree>122</duree>
  <langue>French</langue>
  <prix>9.82</prix>

  Etc.

Prédicats XPath (1/2)

- i^e, dernier, i premiers/derniers éléments

```
//film[1]
//film[last()]
//film[position() < 3]/titre
```

Résultat

```
<titre>Blade runner</titre>
<titre>La grande vadrouille</titre>
```

- Éléments possédant un sous-élément ou attribut donné

```
//film[duree]/titre
```

Résultat

```
<titre>La grande vadrouille</titre>
<titre>Le fabuleux destin d'Amélie Poulain</titre>
<titre>The big Lebowski</titre>
```

```
//film[@sigJeune]
```

Prédicats XPath (2/2)

- Condition sur un élément ou un attribut

```
//film[prix < 15]
//film[@signJeune = "-10" and prix < 5]/titre
```

Résultat

```
<titre>Le fabuleux destin d'Amélie Poulain</titre>
```

- Combinaison de chemins

```
//titre | //prix
```

Résultat

```
<titre>Blade runner</titre><prix>4.79</prix>
<titre>La grande vadrouille</titre><prix>9.82</prix>
<titre>Le fabuleux destin...</titre><prix>4.99</prix>
<titre>The big Lebowski</titre> <prix>9.82</prix>
```

Fonctions XPath

- Fonctions d'accès : data()…
- Fonctions numériques : abs(), floor(), ceiling(), round(), number()…
- Fonctions de chaînes : string-length(), upper-case(), lower-case(), normalize-space(), substring(), substring-after(), replace(), contains()…
- Fonctions temporelles : day-from-date(), year-from-date()…
- Fonctions de séquences : exists(), distinct-values(), reverse(), sort()…
- Fonctions contextuelles : last(), position()…
- Fonctions booléennes : not()…
- Fonctions d'agrégat : count(), sum(), avg(), max(), min()…

Requêtes FLWOR

- For, Let, Where, Order by, Return

- Clause For (1/3) : lie une variable à chaque élément retourné par une expression (itération)

  Exemple

```
for $x in (1 to 3)          <!-- Ceci est un commentaire -->
return <res>{$x}</res>
```

  Résultat

```
<res>1</res>
<res>2</res>
<res>3</res>
```

Clause For (2/3)

Exemple

```
for  $x in (1, 2),
     $y in (10, 20)          (: Ceci est également un commentaire :)
return <x>{$x}</x> <y>{$y}</y>
```

Résultat

```
<x>1</x> <y>10</y>
<x>1</x> <y>20</y>
<x>2</x> <y>10</y>
<x>2</x> <y>20</y>
```

Clause For (3/3)

Exemple

```
for $x at $i in //titre
return <film id="{$i}">{data($x)}</film>
```

Résultat

```
<film id="1">Blade runner</film>
<film id="2">La grande vadrouille</film>
<film id="3">Le fabuleux destin d'Amélie Poulain</film>
<film id="4">The big Lebowski</film>
```

Clause Let

- Clause Let : Assigner une ou plusieurs valeurs à une énumération (pas d'itération)

Exemple

```
let $x := (1 to 5)
return <res>{$x}</res>
```

Résultat

```
<res>1 2 3 4 5</res>
```

Clause Where

- Clause Where : Spécifie une ou plusieurs conditions sur le résultat

Exemple

```
for $x in //film
where $x/prix > 5
return $x/titre
```

Exemple

```
for $x in //film
where $x/@sigJeune = "-12" and $x/prix < 10
return $x/titre
```

Clauses Order by et Return (1/2)

- Clause Order by : Trie le résultat

  Exemple

```
for $x in //film
order by $x/titre
return $x/titre
```

  Exemple

```
for $x in //film
order by $x/@sigJeune, $x/titre descending
return $x/titre
```

- Clause Return : Spécifie le résultat (un ou des fragments XML)

Clause Return (2/2)

- Expressions conditionnelles

Exemple

```
for $x in //film
return   if ($x/@sigJeune="-18")
         then <diffRestr>{data($x/titre)}</diffRestr>
         else <diffLibre>{data($x/titre)}</diffLibre>
```

Résultat

```
<diffLibre>Blade runner</diffLibre>
<diffLibre>La grande vadrouille</diffLibre>
<diffLibre>Le fabuleux destin d'Amélie Poulain</diffLibre>
<diffLibre>The big Lebowski</diffLibre>
```

Fonctions XQuery (les mêmes que XPath !) (1/2)

- Fonctions d'accès : data()…
- Fonctions numériques : abs(), floor(), ceiling(), round(), number()…
- Fonctions de chaînes : string-length(), upper-case(), lower-case(), normalize-space(), substring(), substring-after(), replace(), contains()…
- Fonctions temporelles : day-from-date(), year-from-date()…
- Fonctions de séquences : exists(), distinct-values(), reverse(), sort()…
- Fonctions contextuelles : last(), position()…
- Fonctions booléennes : not()…
- Fonctions d'agrégat : count(), sum(), avg(), max(), min()…

Fonctions XQuery (2/2)

Exemple d'appel à une fonction

```
for $x in //titre
let $titreMAJ := upper-case($x)
return <film>{$titreMAJ}</film>
```

Résultat

```
<film>BLADE RUNNER</film>
<film>LA GRANDE VADROUILLE</film>
<film>LE FABULEUX DESTIN D'AMÉLIE POULAIN</film>
<film>THE BIG LEBOWSKI</film>
```

Clause Group by (XQuery 3)

Regroupement sur un critère

```
for $d in //film
group by $z := $d/@sigJeune
let $prixmoyen := avg($d/prix)
return    <sigJeune value="{$z}">
                <prix_moyen>{$prixmoyen}</prix_moyen>
          </sigJeune>
```

Regroupement multiple

```
for $d in //film
group by $z := $d/@sigJeune, $a := $d/annee
return    <groupe sigJeune="{$z}" annee="{$a}">
                <prix_moyen>{avg($d/prix)}</prix_moyen>
          </groupe>
```

Jointures – Documents exemples (1/3)

```
<?xml version="1.1" encoding="utf-8" ?>      <!-- document 1 : clients.xml -->

<clients>

        <client id="1">
                <nom>Loudcher</nom>
                <prenom>Sabine</prenom>
                <addresse>Bureau K073</addresse>
        </client>

        <client id="2">
                <nom>Bentayeb</nom>
                <prenom>Fadila</prenom>
                <addresse>Bureau K061</addresse>
        </client>

        <client id="3">
                <nom>Darmont</nom>
                <prenom>Jérôme</prenom>
                <addresse>Bureau K067</addresse>
        </client>

</clients>
```

Jointures – Documents exemples (2/3)

```
<?xml version="1.1" encoding="utf-8" ?> <!-- document 2 : produits.xml -->

<produits>

        <produit id="10">
                <nom>Ordinateur</nom>
        </produit>

        <produit id="20">
                <nom>Moniteur</nom>
        </produit>

        <produit id="30">
                <nom>Imprimante</nom>
        </produit>

</produits>
```

Jointures – Documents exemples (3/3)

```
<?xml version="1.1" encoding="utf-8" ?> <!-- document 3 : commandes.xml -->
<commandes>
        <commande cli-id="1" prod-id="10">
                <quantite>3</quantite>
        </commande>
        <commande cli-id="1" prod-id="20">
                <quantite>15</quantite>
        </commande>
        <commande cli-id="2" prod-id="10">
                <quantite>7</quantite>
        </commande>
        <commande cli-id="2" prod-id="30">
                <quantite>10</quantite>
        </commande>
        <commande cli-id="3" prod-id="30">
                <quantite>5</quantite>
        </commande>
</commandes>
```

Jointures de documents XML (1/3)

Exemple

```
for   $c in doc("clients.xml")//client,
      $o in doc("commandes.xml")//commande
where $c/@id = $o/@cli-id
return    <nom>{data($c/nom)}</nom>
          <prenom>{data($c/prenom)}</prenom>
          <qte>data($o/quantite)}</qte>
```

Résultat

```
<nom>Loudcher</nom> <prenom>Sabine</prenom> <qte>3</qte>
<nom>Loudcher</nom> <prenom>Sabine</prenom> <qte>15</qte>
<nom>Bentayeb</nom> <prenom>Fadila</prenom>  <qte>7</res>
<nom>Bentayeb</nom> <prenom>Fadila</prenom>  <qte>10</qte>
<nom>Darmont</nom> <prenom>Jérôme</prenom> <qte>5</qte>
```

Jointures de documents XML (2/3)

Exemple

```
for   $c in doc("clients.xml")//client,
      $o in doc("commandes.xml")//commande,
      $p in doc("produits.xml")//produit
where $c/@id = $o/@cli-id
and $o/@prod-id = $p/@id
return    <nom-cli>{data($c/nom)}</nom-cli>
          <prenom>{data($c/prenom)}</prenom>
          <qte>{data($o/quantite)}</qte>
          <nom-prod>{data($p/nom)}</nom-prod>
```

Résultat

```
<nom-cli>Loudcher</nom-cli> <prenom>Sabine</prenom>  <qte>3</qte>   <nom-prod>Ordinateur</nom-prod>
<nom-cli>Loudcher</nom-cli> <prenom>Sabine</prenom>  <qte>15</qte> <nom-prod>Moniteur</nom-prod>
<nom-cli>Bentayeb</nom-cli> <prenom>Fadila</prenom>  <qte>7</qte>   <nom-prod>Ordinateur</nom-prod>
<nom-cli>Bentayeb</nom-cli> <prenom>Fadila</prenom>  <qte>10</qte> <nom-prod>Imprimante</nom-prod>
<nom-cli>Darmont</nom-cli>  <prenom>Jérôme</prenom> <qte>5</qte>   <nom-prod>Imprimante</nom-prod>
```

Jointures de documents XML (3/3)

Variantes avec les conditions de jointures exprimées en prédicats de chemins

```
for   $c in doc("clients.xml")//client,
      $o in doc("commandes.xml")//commande[@cli-id=$c/@id]
return <nom>{data($c/nom)}</nom> <prenom>{data($c/prenom)}</prenom>
         <qte>{data($o/quantite)}</qte>
```

```
for   $c in //client,
      $p in //produit,
      $o in //commande[@cli-id=$c/@id and @prod-id=$p/@id]
return <nom-cli>{data($c/nom)}</nom-cli> <prenom>{data($c/prenom)}</prenom>
         <qte>{data($o/quantite)}</qte> <nom-prod>{data($p/nom)}</nom-prod>
```

Jointures de documents XML

FLWOR
=
lisibilité

Xpath
=
concision

Plan

- ✔ Introduction
- ✔ Documents XML
- ✔ Langage XQuery