



# Bases de données semi-structurées

Master 2 Humanités numériques  
2023-2024

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/?page_id=3135)



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



<https://social.sciences.re/@darmont> **#hnbds**

- ▶ 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

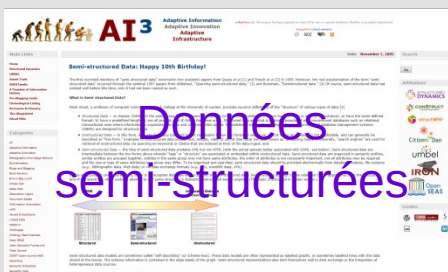
# Données semi-structurées (1/2)

Bases de données

Documents

Années 1990

N° employé	Date embauche	Nom	Prénom	Adresse	Ville	Province	Code postal	Date naissance
0099	12-jan-89	Charut	Eric	1975 rue Duluth	Montréal	QC	H4T 2L9	06-sep-49
0101	20-fev-89	Reys	Kimberly	895 ave. Lakeshore	Donat	QC	H9P 3T2	15-aoû-53
0102	05-mars-89	Lapointe	Claude	308 rue Ste-Francoise	Montréal	QC	H9P 1A6	13-mai-68
0105	14-aoû-89	Bédard	Julie	440 rue Talbot	Laval	QC	H7P 3G8	14-aoû-45
0160	07-mars-89	Ford	Jean-Paul	4380 rue Racicot	Montréal	QC	H9P 2P5	25-juin-59
0230	19-mars-89	Lalancette	Monique	892 rue Victoria	Longueuil	QC	J3P 2T2	02-mai-37
0234	12-avr-89	Phaneuf	Jocelyne	3201 rue Verchères	Côte-Saint-Pierre	QC	J4P 2Y2	02-avr-71
0387	21-avr-89	Provan	Marcel	8085 boul. St-Laurent	Montréal	QC	H3T 2V3	06-mars-56
0390	12-mai-89	Larocque	Hedra	69 rue DuCollege	Ville St-Laurent	QC	H4P 2B8	05-aoû-64
0434	04-juin-89	Comnet	Mark	1213 rue St-Gaulte	St-Hubert	QC	J4T 2L9	28-juin-69
0503	13-juin-89	Aravette	Alan	408 rue Gaudin	Culberrnet	QC	H3T 2G2	25-juin-67
0560	15-aoû-89	Sérand	Math	2575 rue Victoria	St-Lambert	QC	J4L 9P7	03-aoû-74
0728	15-aoû-89	Bédard	Sébastien	3529 rue Rousselle	Ste-Therese	QC	J3P 3V5	14-aoû-51
0839	14-aoû-89	Leblond	Josée	684 rue Montclair	St-Basile	QC	J4L 2Y4	12-aoû-71
1142	10-aoû-89	Favrier	Math	43 rue Harbord	Canada	QC	J2P 2M7	31-aoû-54
1181	14-aoû-89	Bégin	Jake	621 rue Crémieux	Lévis	QC	H9P 3Z7	02-juin-48
1241	07-jan-90	Sim	John	3311 rue	St-Jovite	QC	J3V 9P9	25-aoû-50
1300	01-fev-90	Quinn	Francis	6811 rue des Rangées	Montréal	QC	H9P 2C3	06-aoû-66
1341	04-mars-90	Vermil	Frank	1802 rue	Montréal	QC	J4P 1B2	13-aoû-65
1480	05-avr-90	Quinn	Francis	6811 rue des Rangées	Montréal	QC	H9P 2C3	28-aoû-66
1473	07-aoû-90	Vaudin	Jean-Claude	2689 rue Fleury	Montréal	QC	H4M 1C5	12-aoû-62
1481	01-aoû-90	Wasson	Harrie	38 rue Labadie	Sudbury	QC	J4Z 1T7	07-sep-46
1543	09-aoû-90	Wasson	Harrie	38 rue Labadie	Sudbury	QC	H2T 2V5	30-aoû-46
1589	08-sep-90	Wasson	Harrie	38 rue Labadie	Sudbury	QC	H7P 7P6	03-jan-55
1671	14-aoû-90	Wasson	Harrie	38 rue Labadie	Sudbury	QC	G8B 2G9	17-aoû-62
2200	16-aoû-90	Wasson	Harrie	38 rue Labadie	Sudbury	QC	J4P 1C3	28-sep-43
2312	04-aoû-91	Leblond	Commen	1141 rue Harold	Montréal	QC	H3S 3G3	03-aoû-70
2315	04-aoû-91	Coutard	Sophie	2889 boul. Décarie	Ville St-Laurent	QC	H4T 2Z2	29-juin-69
2320	04-aoû-91	Hopnet	Hughette	305 rue Hutchison	Culberrnet	QC	H3T 2T4	15-aoû-61
2321	01-aoû-91	Milot	Bernard	6967 rue Viger	Montréal	QC	H3V 2M5	10-mai-73
2322	01-aoû-91	Ave	Clara	420 rue Dawson	Montréal	QC	H4L 6L9	18-aoû-67
2340	01-aoû-91	Ouellet	Patricia	7228 boul. St-Joseph	Montréal	QC	H2L 3P5	13-aoû-51
2349	05-jan-92	Achambault	Manon	673 rue Beaubien	Montréal	QC	H4U 2T1	31-aoû-62
2389	04-fev-92	Brenner	Guy	2975 Boulevard	Brossard	QC	J4P 2V1	29-juin-66
2480	04-mars-92	Lagaré	Mirella	592 St-Acme	Montréal	QC	H2T 5C2	26-sep-73
2500	04-mars-92	Phaneuf	France	2687 Christophe-Columbo	Montréal	QC	H9P 2M9	09-aoû-72
3501	04-aoû-92	Trolier	Laurent	1955 rue Villeneuve	Longueuil	QC	J3P 8T3	11-juin-72
3500	04-aoû-92	Wasson	Alexandre	154 rue Dumont	Montréal	QC	J4Z 8M2	14-aoû-69
3554	15-aoû-92	Vissard	François	2102 rue Villeneuve	Montréal	QC	H4T 3B9	30-mars-69
3565	21-aoû-92	Wasson	France	162 rue Ste-Barthe	Verchères	QC	J5C 4J2	20-aoû-72
3561	21-aoû-92	Gagnon	Eric	2235 rue Béliveau	Montréal	QC	H9P 2J9	06-aoû-46
3980	09-aoû-92	Demers	Maria-Claire	1276 rue Van Horne	Culberrnet	QC	H3P 6P5	06-aoû-54
3980	08-aoû-92	Rochette	Cécile	7129 L'Écuyer-Bonaparte	Montréal	QC	H3V 6L3	29-juin-74



Données semi-structurées

**L'ESTA\***, la nouvelle procédure d'autorisation de voyage pour aller ou transférer aux États-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 exempts 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 États-Unis ou avec un transit aux États-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 États-Unis, tout voyageur devra remplir via internet le nouveau formulaire d'inscription en ligne, l'ESTA.

**Attention!** Cette formalité est obligatoire pour :

- TOUT voyageur, qu'il se rende sur le territoire américain ou qu'il y transite.
- Qu'il 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 6 mois et 1 par enfant), accompagné ou non.
- Qu'il soit le mode de transport : avion, train, ferry ou bateau.

**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 est elle gratuite ?**

Non, elle est payante. Elle est destinée aux citoyens américains et à ceux de pays amis.

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.

**Attention!**

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 l'Internet uniquement.

Pour soumettre votre demande d'autorisation ESTA depuis le 1er août 2008, il vous faut 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).

**Attention!**

Si vous n'avez pas accès à Internet, vous devez 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.

Langages de requête

Moteurs de recherche

# 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
- ▶ Données semi-structurées : **autodescriptives**
  - Pages web, documents XML, courriels...

Structurées

Non-structurées

# Exemple de données semi-structurées

- ▶ Nom Jérôme Darmont
- ▶ Courriel [jerome.darmont@univ-lyon2.fr](mailto:jerome.darmont@univ-lyon2.fr)  
[jerome.darmont@msh-lse.fr](mailto:jerome.darmont@msh-lse.fr)
  
- ▶ Courriel [sabine.loudcher@univ-lyon2.fr](mailto: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 être contraintes par un schéma
- Format flexible pour l'interopérabilité
- Permet de voir des données structurées comme semi-structurées (Web)
- Schéma facilement évolutif

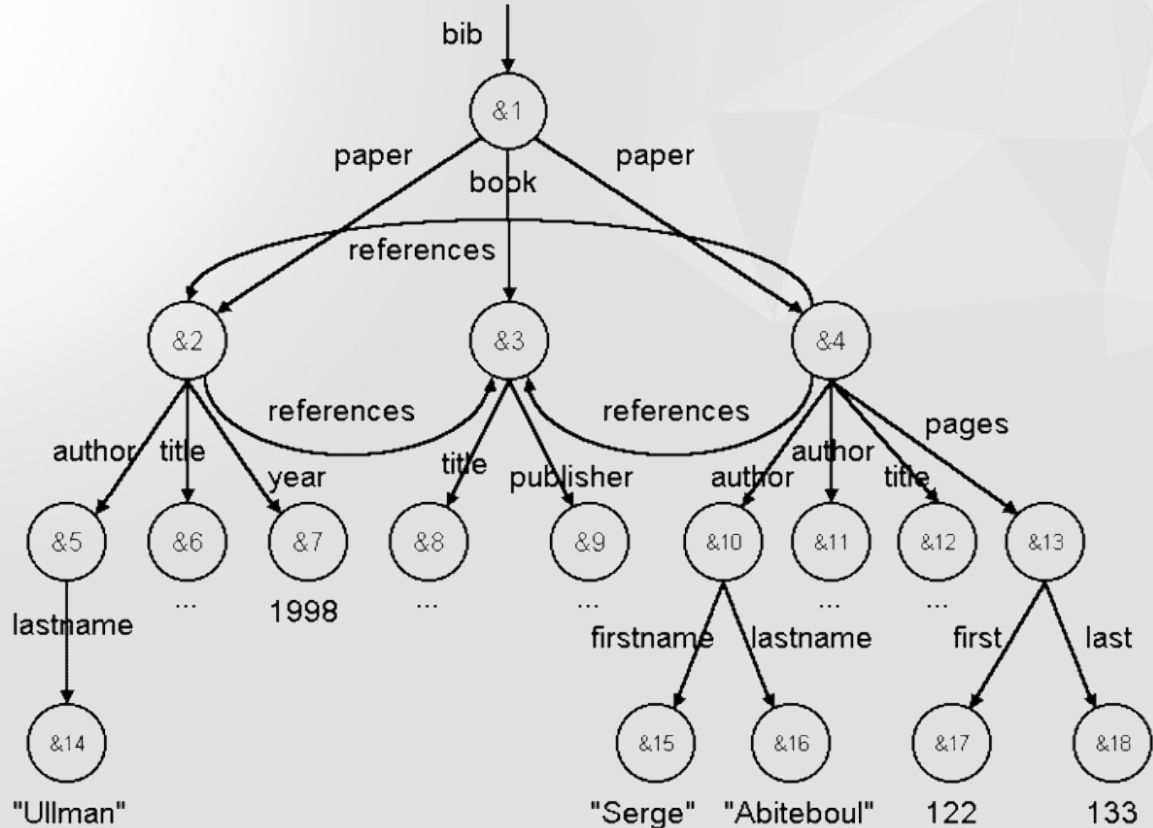
## ▶ Inconvénients

- 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 10<sup>th</sup> Birthday!  
<http://www.mkbergman.com/153/semi-structured-data-happy-10th-birthday/>

## ✔ Introduction

## ▶ Langage XML

## ▶ Langage XQuery

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

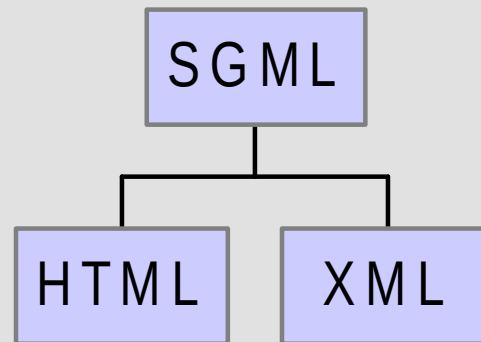


## ▶ XML : Extensible Markup Language

- 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 \_.





- ▶ 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



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



- ▶ Caractères non-autorisés : `<` `&` `]]>`
- ▶ Éléments emboîtés : profondeur non limitée
  - Ex. `<annuaireProfs>`

```
<prof>
  <nom>
    <nom_famille>Zighed</nom_famille>
    <prenom>Abdelkader</prenom>
    <prenom>Djamel</prenom>
  </nom>
</prof>
</annuaireProfs>
```



- ▶ **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** : 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>`

*attribut*    *valeur*    *attribut*    *valeur*

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



## ▶ 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>
```



## ▶ 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





## ▶ 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">`



- ▶ **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 : `(élément1, élément2, ..., élémentn)`
    - **Ex.** `<!ELEMENT prof (nom, courriel, cours)>`
  - Séquence ordonnée : `(élément1 | élément2 | ... | élémentn)`
    - **Ex.** `<!ELEMENT uri (http | ftp | mailto | telnet )>`
  - **Note** : les sous-éléments doivent être définis à leur tour



## ► 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)*>
```



- ▶ **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)>
```



▶ **Déclaration de liste d'attributs** : `<!ATTLIST nom_élt définitions_att>`

- Ex. `<!ATTLIST bureau campus CDATA "Porte des Alpes" bâtiment CDATA #REQUIRED >`

Élément

Attributs

▶ **Définition des attributs**

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



## ► 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">`



- ▶ **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>`



- ▶ **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>`





- ▶ **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...**



## ▶ 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">
```



## ▶ 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 -->

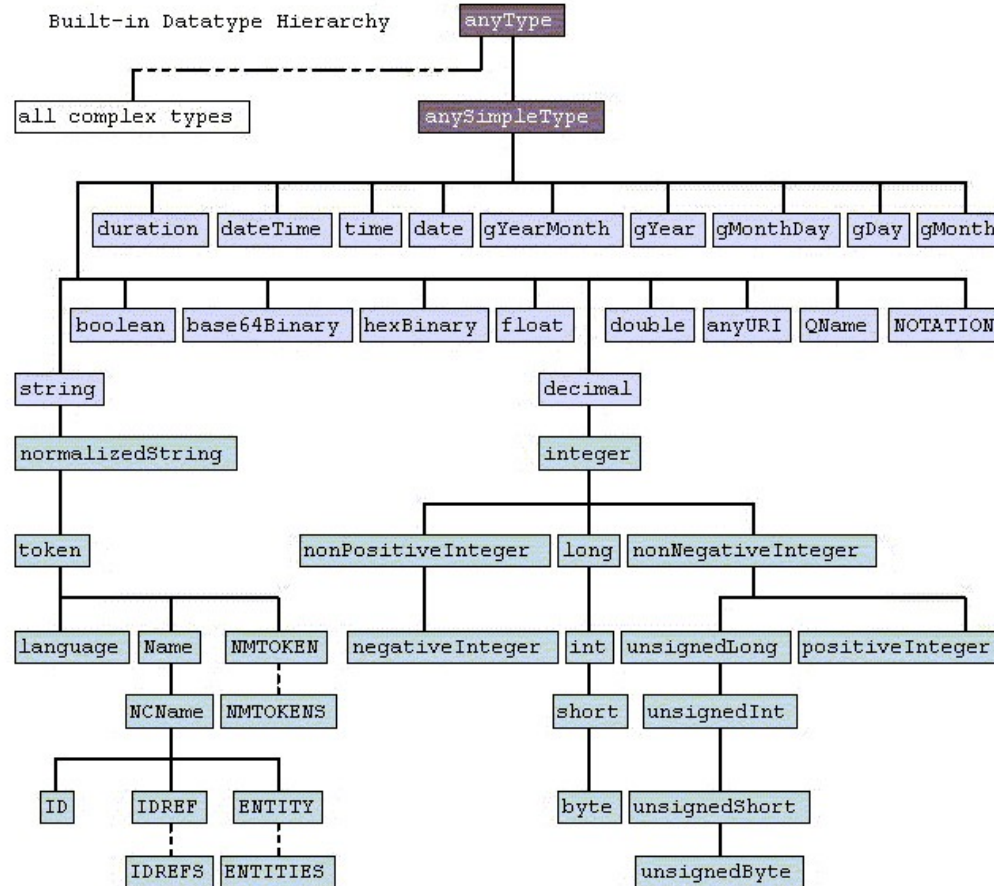
## ▶ Type complexe

```
<xsd:element name="prof">
  <xsd:complexType>
```

<!-- Spécification du type complexe (à suivre) -->

```
</xsd:complexType>
</xsd:element>
```

# Types de données simples





## ▶ 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>
```



## ▶ Tout

<-- Chaque élément apparaît au moins une fois, mais dans n'importe quel ordre -->

```
<xsd:all>  
  <xsd:element name="adresse" type="xsd:string" />  
  <xsd:element name="courriel" type="xsd:string" />  
  <xsd:element name="tel" type="xsd:string" />  
</xsd:all>
```

## ▶ Référence à un élément complexe

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



## ► 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>
```



## ► 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>
```





## ▶ 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 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  $\supset$  XPath 2.0
  - 2017 : XQuery 3.1  $\supset$  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>
```



- ▶ **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>







- ▶ 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>
```



- ▶ Tous les sous-éléments d'un élément  
`doc("films.xml")/catVOD/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.

- ▶ **i<sup>e</sup>, dernier, i premiers/derniers éléments**

```
doc("films.xml")/catVOD/film[1]
```

```
doc("films.xml")/catVOD/film[last()]
```

```
doc("films.xml")/catVOD/film[position() < 3]/titre
```

Résultat

```
<titre>Blade runner</titre>
```

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

- ▶ **Éléments possédant un sous-élément ou attribut donné**

```
doc("films.xml")/catVOD/film[duree]/titre
```

Résultat

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

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

```
<titre>The big Lebowski</titre>
```

```
doc("films.xml")/catVOD/film[@sigJeune]
```



- ▶ **Condition sur un élément ou un attribut**  
doc("films.xml")/catVOD/film[prix < 15]  
doc("films.xml")/catVOD/film[@signJeune = "-10" and prix < 5]/titre  
**Résultat**  
<titre>Le fabuleux destin d'Amélie Poulain</titre>
- ▶ **Combinaison de chemins**  
doc("films.xml")//titre | doc("films.xml")//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>



- ▶ 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>
```



## Exemple

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

## Résultat

```
<res>x = 1 et y = 10</res>  
<res>x = 1 et y = 20</res>  
<res>x = 2 et y = 10</res>  
<res>x = 2 et y = 20</res>
```



## Exemple

```
for $x at $i in doc("films.xml")/catVOD/film/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** : 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** : Spécifie une ou plusieurs conditions sur le résultat

## Exemple

```
for $x in doc("films.xml")/catVOD/film
where $x/prix > 5
return $x/titre
```

## Exemple

```
for $x in doc("films.xml")/catVOD/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 doc("films.xml")/catVOD/film
order by $x/titre
return $x/titre
```

## Exemple

```
for $x in doc("films.xml")/catVOD/film
order by $x/@sigjeune, $x/titre descending
return $x/titre
```

- ▶ **Clause Return** : Spécifie le résultat



## ► Expressions conditionnelles

### Exemple

```
for $x in doc("films.xml")/catVOD/film
return  if ($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 XPath/XQuery (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()`...



## Exemple d'appel à une fonction

```
for $x in doc("films.xml")/catVOD/film/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>
```



## Regroupement sur un critère

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

## Regroupement multiple

```
for $d in /catVOD/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)

XQ



```
<?xml version="1.1" encoding="utf-8" ?>
```

```
<!-- document 1 : clients.xml -->
```

```
<clients>
```

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

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

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



## Exemple

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

## Résultat

```
<res>Loudcher, Sabine : 3</res>  
<res>Loudcher, Sabine : 15</res>  
<res>Bentayeb, Fadila : 7</res>  
<res>Bentayeb, Fadila : 10</res>  
<res>Darmont, Jérôme : 5</res>
```



## 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 = $p/@id  
return <res>{data($c/nom)}, {data($c/prenom)}:  
           {data($o/quantite)} x {data($p/nom)}</res>
```

## Résultat

```
<res>Loudcher, Sabine : 3 x Ordinateur</res>  
<res>Loudcher, Sabine : 15 x Moniteur</res>  
<res>Bentayeb, Fadila : 7 x Ordinateur</res>  
<res>Bentayeb, Fadila : 10 x Imprimante</res>  
<res>Darmont, Jérôme : 5 x Imprimante</res>
```



## 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 <res>{data($c/nom)}, {data($c/prenom)}:  
            {data($o/quantite)}</res>
```

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



## 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 <res>{data($c/nom)}, {data($c/prenom)}:  
            {data($o/quantite)}</res>
```

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



FLWOR  
=  
lisibilité



Xpath  
=  
concision

# Plan

- ✓ Introduction
- ✓ Documents XML
- ✓ Langage XQuery

