# BDBIO - XML et XQuery

Fabien Duchateau

*fabien.duchateau [at] univ-lyon1.fr*

Université Claude Bernard Lyon 1

2023 - 2024


https://perso.liris.cnrs.fr/fabien.duchateau/BDBIO/

# Rappels

Les données d'application sont principalement gérées par des systèmes de gestion de bases de données (SGBD), qui suivent un modèle de données

| **Modèle** | **Sérialisation** | **LDD** | **LMD** | **Exemples SGBD** |
|---|---|---|---|---|
| Relationnel | n-uplets | SQL | SQL | SQLite, MariaDB, PostgreSQL |

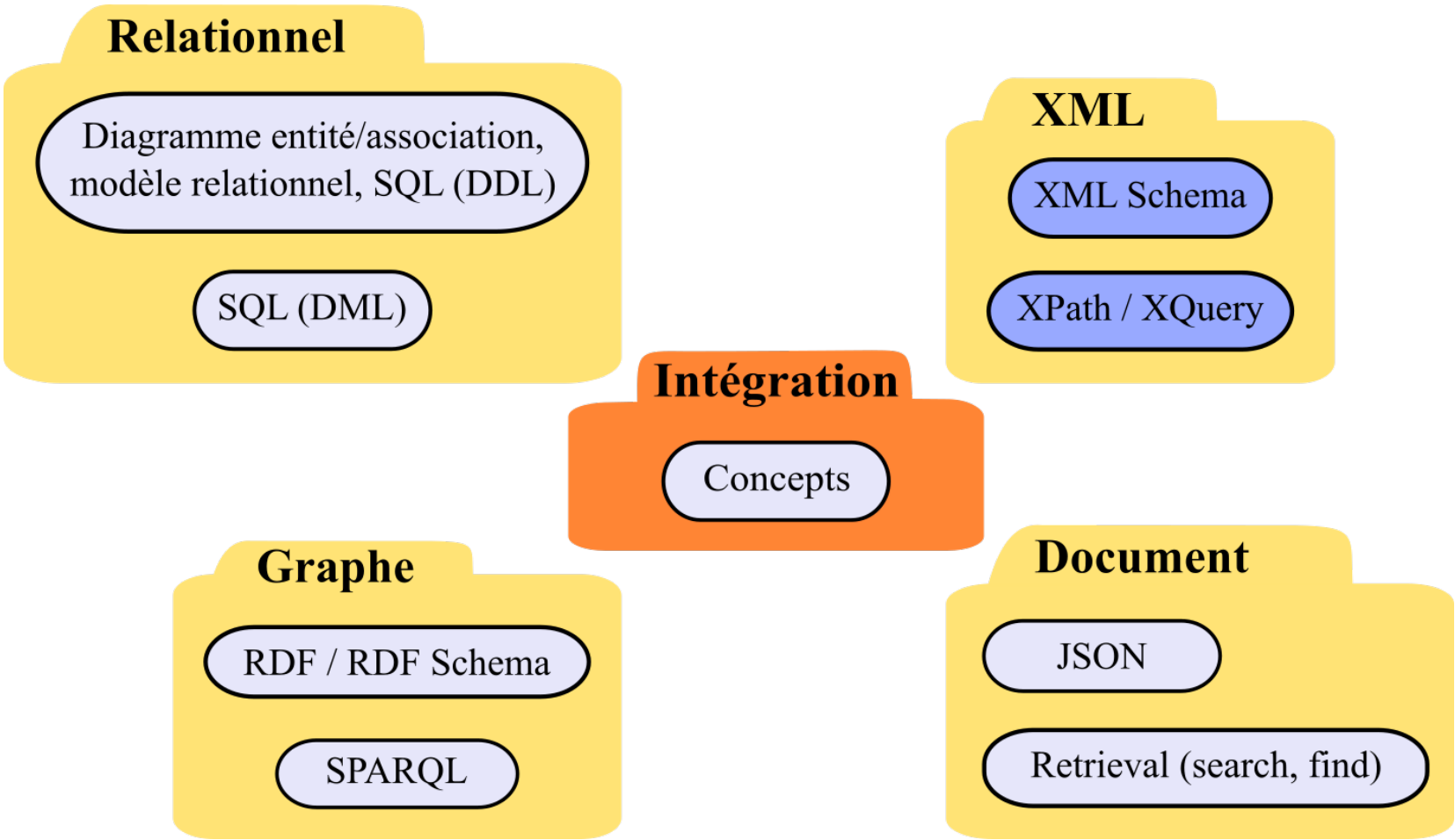
# Rappels

Les données d'application sont principalement gérées par des systèmes de gestion de bases de données (SGBD), qui suivent un modèle de données

| **Modèle** | **Sérialisation** | **LDD** | **LMD** | **Exemples SGBD** |
|---|---|---|---|---|
| Relationnel | n-uplets | SQL | SQL | SQLite, MariaDB, PostgreSQL |

Comment modéliser et interroger des SGBD qui suivent d'autres modèles de données ?

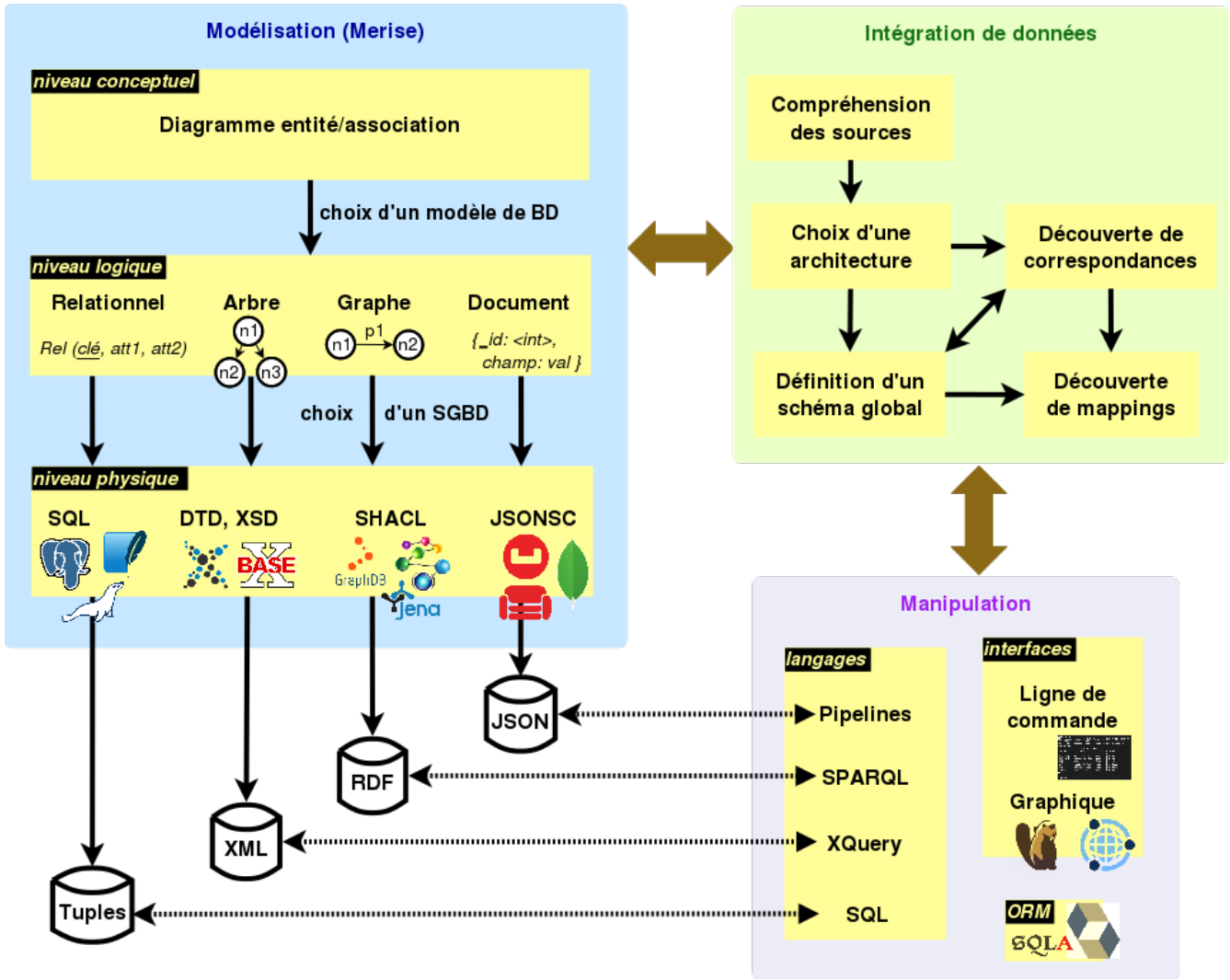
# Positionnement dans BDBIO




---

Ces diapositives utilisent le genre féminin (e.g., chercheuse, développeuses) plutôt que l'écriture inclusive (moins accessible, moins concise, et pas totalement inclusive)

# Relations entre les notions étudiées en BDBIO



# Le langage XML

- XML pour e**X**tensible **M**arkup **L**anguage
- Standard W3C depuis 2008
- Langage de balisage (avec les chevrons)
- Initialement développé pour faciliter l'interopérabilité et le stockage (échange de données, sérialisation)
- Quelques SGBD basés sur le stockage de documents XML

---

http://fr.wikipedia.org/wiki/Extensible_Markup_Language
http://en.wikipedia.org/wiki/Serialization

# Technologies XML

XML regroupe un ensemble de technologies :

- Description d'un document (DTD, XML schema)
- Sélection d'une partie d'un document (XPath)
- Interrogation de document (XQuery)
- Transformation de document (XSLT)
- API pour manipuler un document (DOM pour une représentation en arbre, SAX par déclenchement d'événements)
- ...

---

http://en.wikipedia.org/wiki/XML#Related_specifications
http://en.wikipedia.org/wiki/XML#Programming_interfaces

# SGBD natifs XML

Quelques SGBD orientés document (mouvement NoSQL, post-relationnel) utilisent un document XML comme unité de stockage (logique)

- BaseX
- eXist
- Berkeley DB XML



---

http://en.wikipedia.org/wiki/XML_database
http://basex.org/
http://www.exist-db.org/
http://www.oracle.com/database/berkeley-db/xml.html

# Plan

Concepts de XML

Modélisation en XML

Manipulation avec XPath

Manipulation avec XQuery

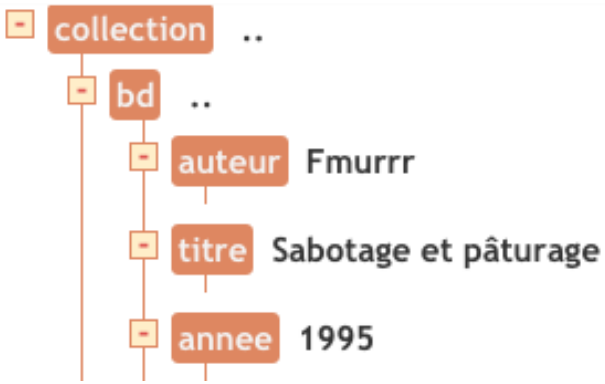
# Définition

Un document XML correspond à un ensemble de données organisées dans un **arbre**

- Format textuel
- Extension de fichier : `.xml`
- Considéré compréhensible à la fois par les humains et les machines
- Syntaxe à base de balises (comme en HTML)
  - balise par paire (`<div>...</div>`)
  - balise orpheline (`<img />`)

# Exemple de document simple

```
<?xml version="1.0" encoding="utf-8"?>
<collection>
    <bd id="1">
        <auteure>Fmurrr</auteure>
        <titre>Sabotage et pâturage</titre>
        <annee>1995</annee>
    </bd>
</collection>
```




*Un simple document avec son entête, un élément `<collection>` qui contient un élément `<bd>`. L'élément `<bd>` est décrit par trois propriétés (éléments et texte)*

# Types de noeuds XML

Un arbre XML possède différents types de noeuds :

- **Racine** = document complet. Elle possède exactement un noeud élément contenant les données
- **Élément** = noeud possédant un nom (balise). Il peut posséder des noeuds filles de tous types (sauf racine)
- **Attribut** = noeud sous forme `nom="valeur"` et qui est rattaché à la balise ouvrante d'un élément. Pour un élément donné, impossible d'avoir deux attribut portant le même nom
- **Texte** entre une balise ouvrante et une fermante
- **Commentaire**

# Syntaxe XML

- Un prologue optionnel (indications pour la lecture du document)

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

- Des éléments imbriqués pouvant contenir des attributs
  - l'ordre entre les noeuds enfants est significatif
  - l'ordre entre les noeuds attributs (d'un même élément) n'est pas important

```
<element attribut="valeur">…</element>
<element attribut="valeur" attribut2="valeur" />
```

# Exemple de syntaxe XML

```xml
<?xml version="1.0" encoding="utf-8"?>
<racine attribut1="valeur1" attribut2="valeur2">
    <!-- commentaire -->
    <element1>
        texte1
    </element1>
    <element2 attribut3="valeur3" />
    <prefixe:element3 attribut4="valeur4">
        texte2
    </prefixe:element3>
</racine>
```

*Un exemple de document XML avec les différents types de noeuds*

# Espaces de nom XML

Un espace de nom est une URI qui permet d'éviter les ambiguïtés sur les noms XML (issus de vocabulaires / schémas différents)

- Nom qualifié = `prefixe:nom`, où `prefixe` est une référence vers une URI
- Nom sans préfixe = nom local, dont l'espace de nommage est celui du schéma
  - s'il n'y a pas d'espace pour le schéma, le nom n'a pas d'espace de nommage

---

http://en.wikipedia.org/wiki/XML_namespace
http://en.wikipedia.org/wiki/Uniform_Resource_Identifier

# Syntaxe des espaces de nom XML

Un espace de nom est précisé par l'attribut `xmlns`

```
<element xmlns:prefix=URI>
<element xmlns=URI>
```

- Un espace de nom défini pour un élément est disponible pour tous les enfants de cet élément
- Sans préfixe, l'espace de nommage est celui par défaut

```
<root xmlns:espace1="uri1" xmlns:espace2="uri2">
    <espace1:element>
        <!-- cette balise element vient du vocabulaire de uri1 -->
    </espace1:element>
    <espace2:element>
        <!-- cette balise element vient du vocabulaire de uri2 -->
    </espace2:element>
</root>
```

# Exemples d'espaces de nom XML

```xml
<?xml version="1.0" encoding="utf-8"?>
<collection xmlns="http://www.bd.db/">
    <bd id="1" xmlns:sch="http://schema.org/">
        <auteure>Fmurrr</auteure>
        <titre>Sabotage et pâturage</titre>
        <annee>1995</annee>
        <sch:numberOfPages>47</sch:numberOfPages>
        <editeur>Dargaud</editeur>
    </bd>
</collection>
```

*Un espace de nom par défaut (pour `<auteur>`, `<titre>`, etc.), et un préfixe `sch` associé à un autre espace de nom uniquement pour l'élément `<bd>`*

# Plan

# Modélisation XML

- Des modèles conceptuels pour XML : XSEM, TreeModel, ERX, EReX, etc.
- Des extensions du diagramme E/A ou du diagramme de classe UML : XER
- Des propositions de conversion de relationnel vers XML

Mais pas de pratique bien établie pour modéliser en XML !

---

Chen, Haitao, and Husheng Liao. *"A survey to conceptual modeling for XML"*. ICCSIT (2010)

M. Franceschet et al. *"From Entity Relationship to XML Schema : a graph-theoretic approach"*. XSym (2009)

J. Fong et al. *"Translating relational schema with constraints into XML schema"*. International Journal of Software Engineering and Knowledge Engineering (2006)

# Schémas pour XML

En XML, un schéma est un ensemble de contraintes structurelles que doit vérifier un document XML pour être valide :

- Noeuds attribut ou noeuds enfants autorisés pour un élément
- Type de valeurs pour les noeuds attribut et noeud texte
- Noeuds attribut ou noeuds enfants obligatoires pour un élément
- ...

**Plusieurs langages pour la définition d'un schéma en XML**

---

http://en.wikipedia.org/wiki/Data_model

# Langages de schéma pour XML

Différents langages de schéma pour des données XML :

- DTD (Document Type Definition)
- XML Schema ou XSD (XML Schema Definition)
- Relax NG : un langage simplifié

Exemples de schémas publics :

- XHTML, RSS
- SVG
- OpenDocument, OpenXML

---

http://schema.org/
http://en.wikipedia.org/wiki/Document_type_definition
http://en.wikipedia.org/wiki/List_of_types_of_XML_schemas
http://en.wikipedia.org/wiki/RELAX_NG

# Document Type Definition (DTD)

- Une DTD précise la structure d'un document XML valide et les balises autorisées (grammaire) ainsi que les nouveaux termes (vocabulaire supplémentaire)
- Une DTD est incluse dans le document XML ou dans un fichier externe (extension `.dtd`)
- Syntaxe spécifique
- Limitations par rapport à d'autres schémas (e.g., type de données, contrainte complexes)

---

http://en.wikipedia.org/wiki/Document_type_definition
http://www.w3.org/TR/REC-xml/#sec-prolog-dtd
http://www.w3schools.com/xml/xml_dtd_intro.asp

# Exemple de DTD

```
<!DOCTYPE collection [
<!ELEMENT collection (serie*)>
<!ELEMENT serie (nom, tome*)>
<!ATTLIST serie id ID #REQUIRED>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT tome (auteure+, titre, annee?, genre*)>
<!ATTLIST tome numero CDATA #IMPLIED>
<!ELEMENT auteure (#PCDATA)>
<!ELEMENT titre (#PCDATA)>
<!ELEMENT annee (#PCDATA)>
<!ELEMENT genre (#PCDATA)>
]>
```

*L'élément racine est `collection`, et contient 0 ou plusieurs `séries`. Une série possède un seul `nom`, et 0 ou plusieurs `tomes`. La série a également un attribut obligatoire `id` de type `ID`. Le nom de la série est une chaîne de caractère (#PCDATA). Un tome contient au moins une `auteure`, un `titre`, une `année` optionnelle et 0 ou plusieurs genres. Un tome possède aussi un attribut `numero` optionnel de type chaîne (#CDATA), etc.*

# XML Schema (XSD)

- XML Schema Definition
- Extension de fichier : `.xsd`
- Par rapport au langage DTD :
  - XSD est un fichier XML, donc facilite le "parsing"
  - XSD permet le typage des données et ajoute de nouvelles contraintes
  - possibilité de référencer d'autres schémas
  - langage assez complexe

---

http://en.wikipedia.org/wiki/XML_Schema_(W3C)
http://www.w3.org/TR/xmlschema-2/

# Exemples d'éléments simples en XML Schema

```
<xsd:element name="dateNaissance" type="xsd:date" />
<xsd:element name="dateParution" type="xsd:gYearMonth" />
<!-- ... -->
<dateNaissance>2000-01-31</dateNaissance><!-- 31/01/2000 -->
<dateParution>2004-06</dateParution><!-- juin 2004 -->
```

*Définition de deux éléments typés (lignes 80 et 81) et exemples d'instance pour ces définitions (lignes 83 et 84)*

```
<xsd:attribute name="numéro" type="xsd:ID" />
<xsd:attribute name="tomes" type="xsd:IDREFS" />
<!-- ... -->
<bd numero="11">Sabotage et pâturage</bd>
<bd numero="13">Cheptel maudit</bd>
<serie tomes="11 13">Le génie des alpages</serie>
```

*Définition d'attributs typés (lignes 87 et 88) et exemples d'instances pour ces définitions (lignes 90 à 92). Les éléments qui sont autorisés à utiliser un attribut sont définis par des types complexes.*

# Exemple d'élément complexe en XML Schema

Un élément complexe avec contenu texte est composé d'un texte (valeur entre les balises) et d'éventuels attributs

```
<xsd:element name="titre">
    <xsd:complexType>
        <xsd:simpleContent>
            <xsd:extension base="xsd:string" />
        </xsd:simpleContent>
    </xsd:complexType>
</xsd:element>
<!-- ... -->
<titre>Sabotage et pâturage</titre>
```

*Définition d'un élément complexe avec un texte de type chaine (lignes 111 à 117) et exemples d'instance pour cette définition (ligne 119)*

# Exemple d'élément complexe en XML Schema (2)

Un élément complexe avec contenu éléments est composé d'un ou plusieurs éléments et d'éventuels attributs

```
<xsd:element name="bd">
    <xsd:complexType>
        <xsd:all>
            <xsd:element name="auteure" type="xsd:string"/>
            <xsd:element name="titre" type="xsd:string"/>
            <xsd:element name="annee" type="xsd:gYear"/>
        </xsd:all>
        <xsd:attribute name="numero" type="xsd:ID"/>
    </xsd:complexType>
</xsd:element>
```

*Définition d'un élément complexe "bd" contenant trois éléments (obligatoires et uniques) et un attribut*

# Lier un document XML à un schéma

```xml
<?xml version="1.0" encoding="utf-8"?>
<collection xmlns="http://www.bd.db/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.bd.db/unSchema.xsd">
<!-- ... -->
</collection>
```

*Document XML contenant un lien vers un schéma XML avec espace de nom*

```xml
<?xml version="1.0" encoding="utf-8"?>
<collection xmlns="http://www.bd.db/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="unSchema.xsd">
<!-- ... -->
</collection>
```

*Document XML contenant un lien vers un schéma XML sans espace de nom*

# Exemple de modélisation - imbrication totale

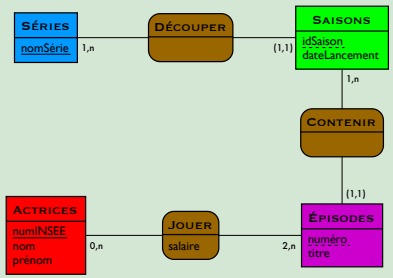
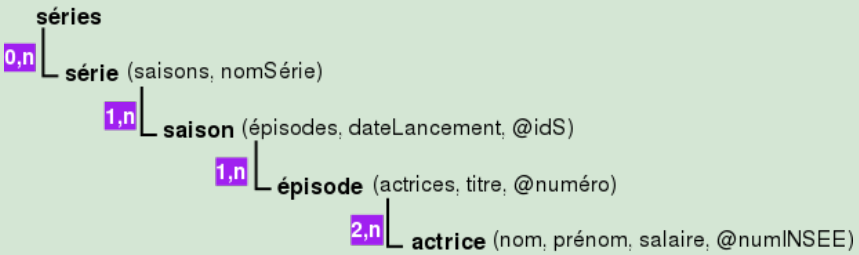


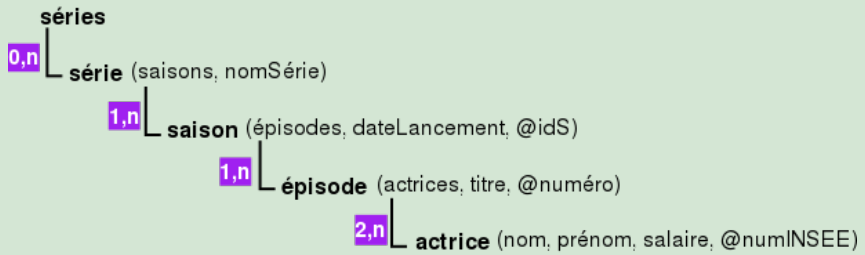

*Diagramme E/A (niveau conceptuel)*

# Exemple de modélisation - imbrication totale




*Représentation libre (niveau logique)*

# Exemple de modélisation - imbrication totale

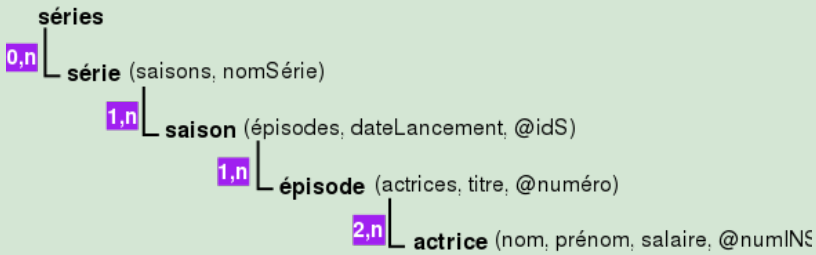



*Représentation libre (niveau logique)*

```
<!DOCTYPE collection [
<!ELEMENT séries (série*)>
<!ELEMENT série (nomSérie, saisons)>
<!ELEMENT nomSérie (#PCDATA)>
<!ELEMENT saisons (saison+)>
<!ELEMENT saison (dateLancement, épisodes?)>
<!ATTLIST saison idS ID #REQUIRED>
<!ELEMENT dateLancement (#PCDATA)>
<!ELEMENT épisodes (épisode+)>
<!ELEMENT épisode (titre, actrices)>
<!ATTLIST épisode numéro CDATA #REQUIRED>
<!ELEMENT titre (#PCDATA)>
<!ELEMENT actrices (actrice+)>
<!ELEMENT actrice (nom, prénom, salaire)>
<!ATTLIST actrice numINSEE ID #REQUIRED>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT prénom (#PCDATA)>
<!ELEMENT salaire (#PCDATA)>
]>
```

*DTD (niveau physique)*

# Exemple de modélisation - imbrication totale




*Représentation libre (niveau logique)*

```
<!DOCTYPE collection [
<!ELEMENT séries (série*)>
<!ELEMENT série (nomSérie, saisons)>
<!ELEMENT nomSérie (#PCDATA)>
<!ELEMENT saisons (saison+)>
<!ELEMENT saison (dateLancement, épisodes?)>
<!ATTLIST saison idS ID #REQUIRED>
<!ELEMENT dateLancement (#PCDATA)>
<!ELEMENT épisodes (épisode+)>
<!ELEMENT épisode (titre, actrices)>
<!ATTLIST épisode numéro CDATA #REQUIRED>
<!ELEMENT titre (#PCDATA)>
<!ELEMENT actrices (actrice+)>
<!ELEMENT actrice (nom, prénom, salaire)>
<!ATTLIST actrice numINSEE ID #REQUIRED>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT prénom (#PCDATA)>
<!ELEMENT salaire (#PCDATA)>
]>
```

*DTD (niveau physique)*

```
<séries>
    <série>
        <nomSérie>Love, death and
        ↪ robots</nomSérie>
        <saisons>
            <saison idS="123">
                <dateLancement>2019</dateLancement>
                <épisodes>
                    <épisode numéro="1">
                        <titre>Sonnie's
                        ↪ Edge</titre>
                        <actrices>
                            <actrice
                            ↪ numINSEE="12345">
                                <nom>Sadler</nom>
                                <prénom>Helen<prénom>
                                <salaire>3000</salaire>
                            </actrice>
                            ...
                        </actrices>
                    </épisode>
                    ...
                </épisodes>
            </saison>
            ...
        </saisons>
    </série>
    ...
</séries>
```

*Exemple de document (données)*

# Exemple de modélisation - concept Actrice hors de Série

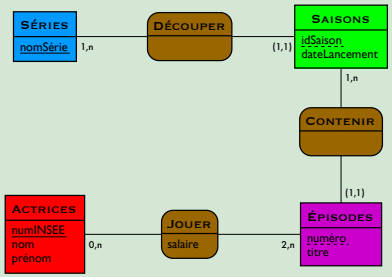
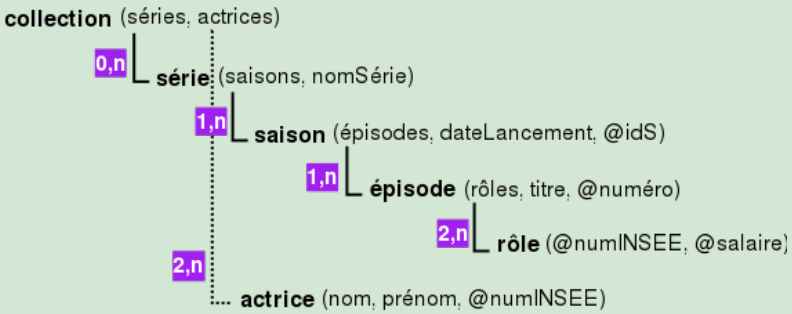


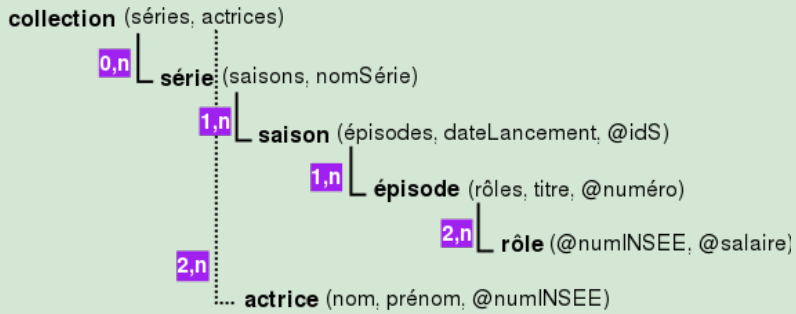

*Diagramme E/A (niveau conceptuel)*

# Exemple de modélisation - concept Actrice hors de Série




*Représentation libre (niveau logique)*

# Exemple de modélisation - concept Actrice hors de Série

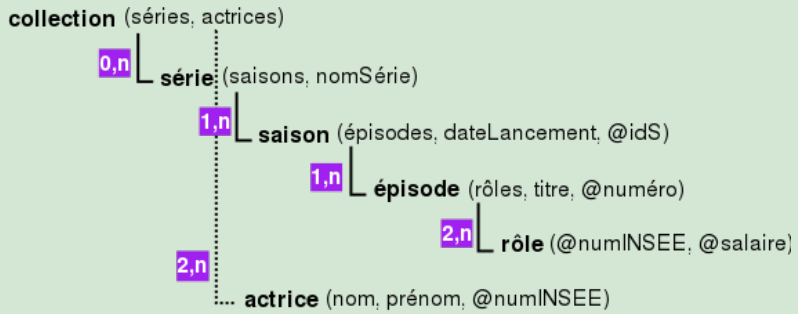



*Représentation libre (niveau logique)*

```
<!DOCTYPE collection [
<!ELEMENT bd (séries, actrices)>
<!ELEMENT séries (série*)>
<!ELEMENT série (nomSérie, saisons)>
<!ELEMENT nomSérie (#PCDATA)>
<!ELEMENT saisons (saison+)>
<!ELEMENT saison (dateLancement, épisodes?)>
<!ATTLIST saison idS ID #REQUIRED>
<!ELEMENT dateLancement (#PCDATA)>
<!ELEMENT épisodes (épisode+)>
<!ELEMENT épisode (titre, roles)>
<!ATTLIST épisode numéro CDATA #REQUIRED>
<!ELEMENT titre (#PCDATA)>
<!ELEMENT roles (role+)>
<!ELEMENT role EMPTY>
<!ATTLIST role numINSEE IDREF #REQUIRED>
<!ATTLIST role salaire CDATA #IMPLIED>
<!ELEMENT actrices (actrice+)>
<!ELEMENT actrice (nom, prénom)>
<!ATTLIST actrice numINSEE ID #REQUIRED>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT prénom (#PCDATA)>
]>
```

*DTD (niveau physique)*

# Exemple de modélisation - concept Actrice hors de Série




*Représentation libre (niveau logique)*

```
<!DOCTYPE collection [
<!ELEMENT bd (séries, actrices)>
<!ELEMENT séries (série*)>
<!ELEMENT série (nomSérie, saisons)>
<!ELEMENT nomSérie (#PCDATA)>
<!ELEMENT saisons (saison+)>
<!ELEMENT saison (dateLancement, épisodes?)>
<!ATTLIST saison idS ID #REQUIRED>
<!ELEMENT dateLancement (#PCDATA)>
<!ELEMENT épisodes (épisode+)>
<!ELEMENT épisode (titre, roles)>
<!ATTLIST épisode numéro CDATA #REQUIRED>
<!ELEMENT titre (#PCDATA)>
<!ELEMENT roles (role+)>
<!ELEMENT role EMPTY>
<!ATTLIST role numINSEE IDREF #REQUIRED>
<!ATTLIST role salaire CDATA #IMPLIED>
<!ELEMENT actrices (actrice+)>
<!ELEMENT actrice (nom, prénom)>
<!ATTLIST actrice numINSEE ID #REQUIRED>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT prénom (#PCDATA)>
]>
```

*DTD (niveau physique)*

```
<bd>
    <séries>
        <série>
            <nomSérie>Love, death and
            ↪ robots</nomSérie>
            <saisons>
                <saison idS="123">
                    <dateLancement>2019</dateLancement>
                    <épisodes>
                        <épisode numéro="1">
                            <titre>Sonnie's
                            ↪ Edge</titre>
                            <roles>
                                <role
                                ↪ numINSEE="12345"
                                ↪ salaire="3000"
                                ↪ />
                                ...
                            </roles>
                        </épisode>
                        ...
                    </épisodes>
                </saison>
                ...
            </saisons>
        </série>
        ...
    </séries>
    <actrices>
        <actrice numINSEE="12345">
            <nom>Sadler</nom>
            <prénom>Helen<prénom>
        </actrice>
        ...
    </actrices>
</bd>
```

*Exemple de document (données)*

# En résumé

- Modèles conceptuels pour XML peu utilisés
- Définition d'un schéma pour un document XML :
  - DTD (grammaire et vocabulaire, mais pas de contraintes de validité)
  - XML Schema (XSD), plus avancé que DTD
- Plusieurs manières d'écrire un schéma (différents langages, mais aussi différents choix de modélisation)
- En pratique, de nombreux documents XML ne possèdent pas de schéma associé

# Plan

# Généralités

XML Path Language (XPath) :

- Sélection de morceaux d'un document XML
- Utilisé par d'autres langages comme XQuery ou XSLT ou par des bibliothèques (Python, Java, etc.)
- Expression XPath : chemins dans l'arbre XML pour sélectionner les noeuds intéressants
  - chemins similaires à la syntaxe de chemin de fichiers sous Unix
- Une version compacte (mais limitée) et une version complète (mais verbeuse)

---

http://fr.wikipedia.org/wiki/XPath
http://www.w3.org/TR/xpath-3/

# Exemples

```xml
<?xml version="1.0" encoding="utf-8"?>
<collection>
    <bd id="1">
        <auteure>Fmurrr</auteure>
        <titre>Sabotage et pâturage</titre>
        <annee>1995</annee>
    </bd>
</collection>
```

*Quelques exemples d'expressions XPath :*

```
/collection/bd
//auteure
collection/bd/annee[text() = 1995]/parent::node()
```

# Syntaxe (complète)

`axe::test[prédicat]/ … /axe::test[prédicat]`

- Expression XPath = suite d'étapes
- Étape = `axe::test[prédicat]`, avec le prédicat optionnel
- Évaluation d'une expression XPath :
  - soit à partir du noeud courant
  - soit à partir de la racine si l'expression commence par un slash (e.g., `/collection`)

Chaque étape de l'expression XPath modifie l'ensemble des noeuds sélectionnés

---

La syntaxe compacte ne respecte pas cette syntaxe (utilisation de raccourcis)

# Axes

Un axe indique la direction de navigation dans l'arbre

Axes en arrière (remontent dans l'arbre) :

- `ancestor` pour les ancêtres du noeud courant
- `ancestor-or-self`
- `parent`
- `preceding` pour les noeuds qui précédent le noeud courant, hors ancêtres
- `preceding-sibling` pour les noeuds de même parent et qui précédent le noeud courant

---

http://en.wikipedia.org/wiki/XPath#Axis_specifiers
http://www.w3.org/TR/xpath-3/#axes

# Axes (2)

Axes en avant (descendent dans l'arbre) :

- `attribute` pour les attributs du noeud courant
- `child` pour les enfants du noeud courant
- `descendant` pour les descendants du noeud courant
- `descendant-or-self`
- `following` pour les noeuds qui suivent le noeud courant, hors descendants
- `following-sibling` pour les noeuds de même parent et qui suivent le noeud courant
- `self` pour le noeud courant

# Tests de noeud

Un test de noeud sélectionne, parmi les noeuds désignés par l'axe, ceux avec un nom ou un type donné

- *nom* pour les éléments avec le nom donné
- `*` pour tous les noeuds élément
- `node()` pour tous les noeuds
- `attribute()` pour tous les éléments de type attribut
- `text()` pour tous les éléments de type texte
- ...

---

http://en.wikipedia.org/wiki/XPath#Node_tests
http://www.w3.org/TR/xpath-3/#dt-node-test

# Prédicat

Un prédicat filtre les noeuds sélectionnés en évaluant une expression booléenne (similaire au WHERE de SQL)

- Vrai si l'expression retourne un résultat non vide
- L'expression peut :
  - combiner différentes conditions (`and`, `or`)
  - utiliser des fonctions booléennes
  - être un chemin (évalué à partir du noeud à tester)
  - utiliser des fonctions non booléennes (avec éventuellement un chemin comme argument)

---

http://en.wikipedia.org/wiki/XPath#Predicates
http://www.w3.org/TR/xpath-3/#dt-predicate

# Quelques abréviations de syntaxe

- `..` ↔ `parent::node()`
- `.` ↔ `self::node()`
- `//` ↔ `descendant-or-self::node()/`
- `balise` ↔ `child::balise`
- `@att` ↔ `attribute::att`
- `[X]` ↔ `[fn:position() = X]`
- ...

# Exemples XPath

**Tous les titres de tome (sans la balise titre) :**

```
//titre/text()
```

**Les tomes avec un genre :**

```
//tome[genre]
```

**Le nom des séries avec un tome 13 :**

```
//serie[tome/@numero = 13]/nom
```

**Les tomes avec un genre et un numéro supérieur ou égal à 1 :**

```
//tome[genre and @numero >= 1]
```

**Le titre des tomes dont le numéro est inférieur à 12 :**

```
//tome[@numero <= 12]/titre
```

```
<collection>
  <serie id="1">
    <nom>Le génie des alpages</nom>
    <tome numero="11">
      <auteure>Fmurrr</auteure>
      <titre>Sabotage et pâturage</titre>
      <annee>1995</annee>
    </tome>
    <tome numero="13">
      <auteure>Fmurrr</auteure>
      <titre>Cheptel maudit</titre>
      <annee>2004</annee>
    </tome>
  </serie>
  <serie id="2">
    <nom>Tu mourras moins bête</nom>
    <tome numero="1">
      <auteure>Marion Montaigne</auteure>
      <titre>La science, c'est pas du cinéma!</titre>
      <annee>2011</annee>
      <genre>Humour</genre>
    </tome>
  </serie>
</collection>
```

# Fonctions

- `fn:round(num)` arrondit *num* à l'entier le plus proche
- `fn:position()` retourne la position d'un noeud
- `fn:not(expr)` retourne vrai si *expr* est fausse
- `[fn:contains(foin, aiguille)]` est vrai si la chaine *foin* contient la chaine *aiguille*
- `fn:count(elem, elem, …)` retourne le nombre de noeuds
- …

---

http://www.w3schools.com/xml/xsl_functions.asp

# Exemples de fonctions XPath

**Le nombre de séries :**

```
/collection/fn:count(node())
ou fn:count(//serie)
```

**Les tomes en seconde position :**

```
//tome[2]
```

**Les séries dont un noeud textuel commence par "T" :**

```
//serie[descendant::node()
[fn:starts-with(child::text(), "T")]]
```

**Les séries dont aucun noeud textuel ne commence par "T" :**

```
//serie[fn:not(descendant::node()[fn:starts-with(
child::text(), "T")])]
```

```
<collection>
  <serie id="1">
    <nom>Le génie des alpages</nom>
    <tome numero="11">
      <auteure>Fmurrr</auteure>
      <titre>Sabotage et pâturage</titre>
      <annee>1995</annee>
    </tome>
    <tome numero="13">
      <auteure>Fmurrr</auteure>
      <titre>Cheptel maudit</titre>
      <annee>2004</annee>
    </tome>
  </serie>
  <serie id="2">
    <nom>Tu mourras moins bête</nom>
    <tome numero="1">
      <auteure>Marion Montaigne</auteure>
      <titre>La science, c'est pas du cinéma!</titre>
      <annee>2011</annee>
      <genre>Humour</genre>
    </tome>
  </serie>
</collection>
```

# Plan

# Généralités

XML Query (XQuery) :

- Langage de manipulation pour des documents XML (utilisé par les SGBD XML)
- Fabrication de (morceaux de) documents XML à partir de données XML
- Standard W3C, basé sur XPath
- Requêtes FLWOR contenant 5+ clauses

---

https://www.w3.org/TR/2017/REC-xquery-31-20170321/#id-flwor-expressions

https://fr.wikipedia.org/wiki/XQuery

https://www.w3schools.com/xml/xquery_intro.asp

https://www.martinbroadhurst.com/open-source-xquery-implementations.html

# Syntaxe FLWOR

**for** $var **in** chemin_xpath ou intervalle
**let** $$var_1$ := $val_1$, ..., $$var_n$ := $val_n$
**where** condition
**order by** expression
**return** expression_xquery

- `for` pour sélectionner une séquence de noeuds, chaque élément étant stocké dans la variable $var
- `let` pour lier une séquence à une variable (sans itération)
- `where` pour filtrer les noeuds (prédicats reliés par *and*/*or*)
- `order by` pour trier les noeuds (*ascending*, *descending*)
- `return` pour retourner un résultat par noeud (seule clause obligatoire)

---

http://www.w3schools.com/xml/xquery_select.asp

# Comparaisons et fonctions en XQuery

- Deux symboles pour les mêmes comparaisons :
  - =, !=, <, <=, >, >= (comparaison vérifiée au moins une fois)
  - eq, ne, lt, le, gt, ge (comparaison vérifiée obligatoirement une seule fois)
- Possibilité d'un `if - then - else` dans la clause `return`
- Mêmes fonctions et opérateurs que XPath : *fn :round()*, *fn :contains()*, *fn :substring()*, etc.

```
for $x in doc("collection.xml")/collection/serie/tome
where $x/annee > 2000 and contains($x/auteure/text(), "g")
order by $x/titre
return $x/titre
```

*La fonction `doc()` permet d'ouvrir un document XML, elle est ici suivie directement d'une expression XPath*

http://www.w3schools.com/xml/xsl_functions.asp

# Exemple de requête FLWOR

```
for $to in //tome
let $ti := $to/titre
where $to/@numero > 5
order by $ti descending
return
    <album>{$to/@numero}
        {$ti}
        <serie>{$to/../nom/text()}</serie>
    </album>
```

```
<collection>
  <serie id="1">
    <nom>Le génie des alpages</nom>
    <tome numero="11">
      <auteure>Fmurrr</auteure>
      <titre>Sabotage et pâturage</titre>
      <annee>1995</annee>
    </tome>
    <tome numero="13">
      <auteure>Fmurrr</auteure>
      <titre>Cheptel maudit</titre>
      <annee>2004</annee>
    </tome>
  </serie>
  <serie id="2">
    <nom>Tu mourras moins bête</nom>
    <tome numero="1">
      <auteure>Marion Montaigne</auteure>
      <titre>La science, c'est pas du cinéma!</titre>
      <annee>2011</annee>
      <genre>Humour</genre>
    </tome>
  </serie>
</collection>
```

# Exemple de requête FLWOR

```
for $to in //tome
let $ti := $to/titre
where $to/@numero > 5
order by $ti descending
return
    <album>{$to/@numero}
        {$ti}
        <serie>{$to/../nom/text()}</serie>
    </album>
```

```
<collection>
  <serie id="1">
    <nom>Le génie des alpages</nom>
    <tome numero="11">
      <auteure>Fmurrr</auteure>
      <titre>Sabotage et pâturage</titre>
      <annee>1995</annee>
    </tome>
    <tome numero="13">
      <auteure>Fmurrr</auteure>
      <titre>Cheptel maudit</titre>
      <annee>2004</annee>
    </tome>
  </serie>
  <serie id="2">
    <nom>Tu mourras moins bête</nom>
    <tome numero="1">
      <auteure>Marion Montaigne</auteure>
      <titre>La science, c'est pas du cinéma!</titre>
      <annee>2011</annee>
      <genre>Humour</genre>
    </tome>
  </serie>
</collection>
```

```
<album numero="11">
  <titre>Sabotage et pâturage</titre>
  <serie>Le génie des alpages</serie>
</album>
<album numero="13">
  <titre>Cheptel maudit</titre>
  <serie>Le génie des alpages</serie>
</album>
```

*Le résultat (clause `return`) mélange des balises XML avec des expressions XQuery pour construire un nouveau document*

# Bilan

- Syntaxe d'un document XML
- Définition d'un schéma pour un document avec XSD (XML Schema)
- Sélection de chemins dans un document avec XPath
- Interrogation de document avec XQuery