XSLT pour les nuls

Lou Burnard Consulting

mai 2011

Objectifs

Ceci n'est pas une formation complète! son objectif est de ...

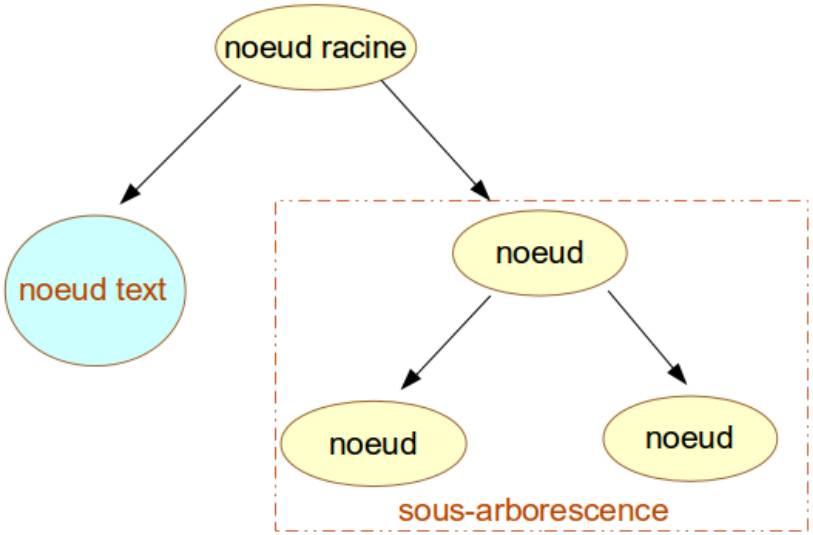
- vous donner un avant-goût des possibilités offertes par les normes XSLT et XPath
- surtout dans le domaine de traitement des documents 'XML-TEI', principalement donc sortis des sciences humaines
- aborder les concepts fondamentaux et les usages les plus répandus du norme XSLT

XSL: un ensemble de normes complémentaires

- XPath: un syntaxe normalisé pour définir et accéder aux sous-parties d'une arborescence XML
- XSLT: un norme informatique pour la transformation des arborescences XML
- XSL FO: un vocabulaire XML pour la description d'affichage des pages

Tous les trois développés et maintenus par le W3C, comme le norme XML.

C'est quoi une arborescence ?



- un ensemble de *nœuds*, organisé de manière hiérarchique
- chaque nœud porte un *identifiant générique* (son "type")
- il y a un seul *nœud racine* qui contient (ou domine) tous les autres
- chaque nœud peut contenir (ou dominer)
  - un sous-arborescence
  - ou un morceau de texte

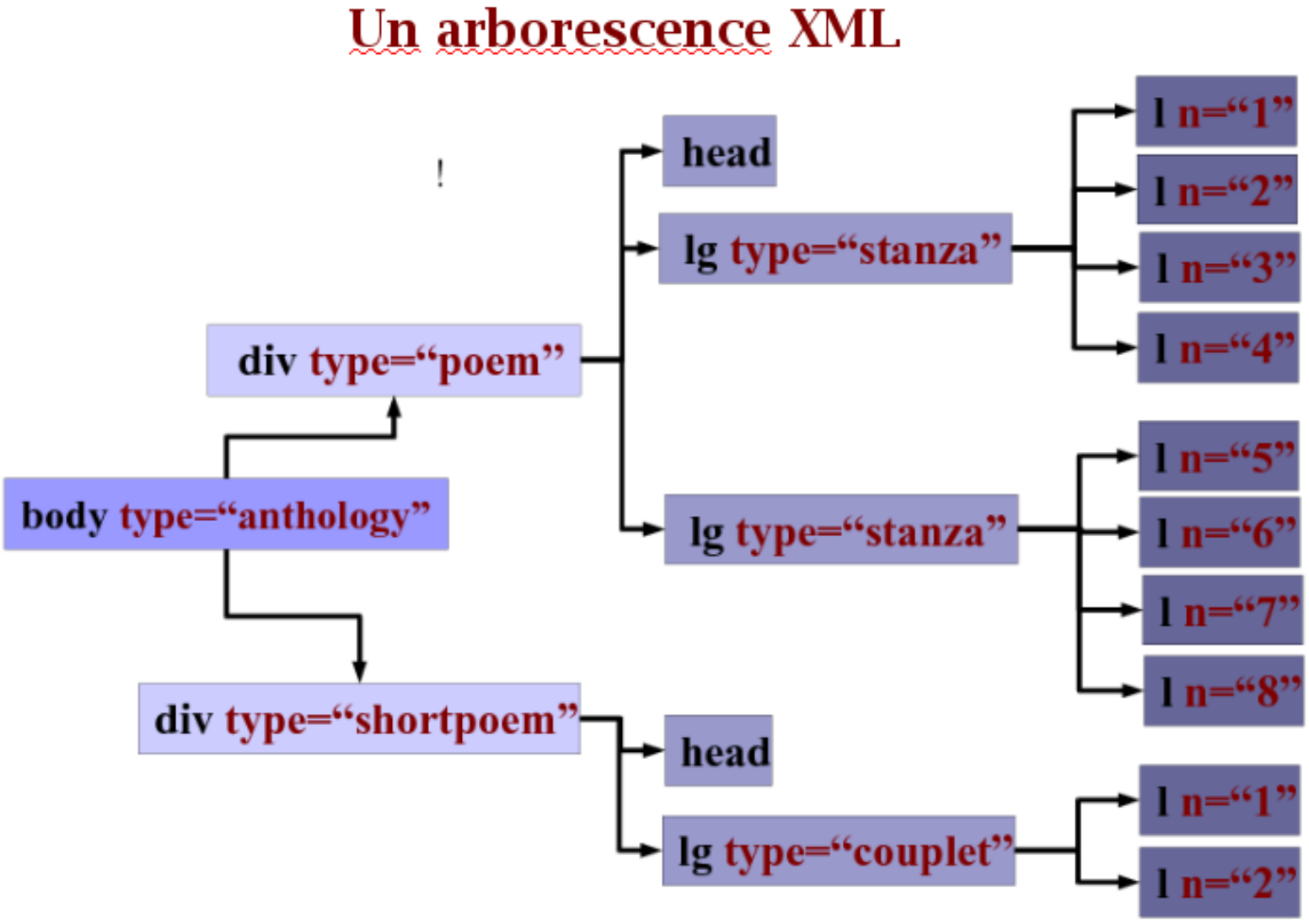
Arborescence XML

- chaque nœud correspond à un élément identifié
- les attributs d'un élément constitue une sous-arborescence associée au nœud
- chaque attribut comporte un *nom* et un *valeur*

Exemple texte XML ...

```
<body type="anthology">
 <div type="poem">
  <head>The SICK ROSE </head>
  <lg type="stanza">
   <l n="1">O Rose thou art sick.</l>
   <l n="2">The invisible worm,</l>
   <l n="3">That flies in the night </l>
   <l n="4">In the howling storm:</l>
  </lg>
  <lg type="stanza">
   <l n="5">Has found out thy bed </l>
   <l n="6">Of crimson joy:</l>
   <l n="7">And his dark secret love </l>
   <l n="8">Does thy life destroy.</l>
  </lg>
 </div>
 <div type="shortpoem">
  <head>Queen Anne's tipple</head>
  <lg type="couplet">
   <l n="1">Here thou Great Anna whom three realms obey</l>
   <l n="2">Doth sometimes council take, and sometimes
tea.</l>
  </lg>
 </div>
</body>
```

.. ou, en forme d'arborescence:



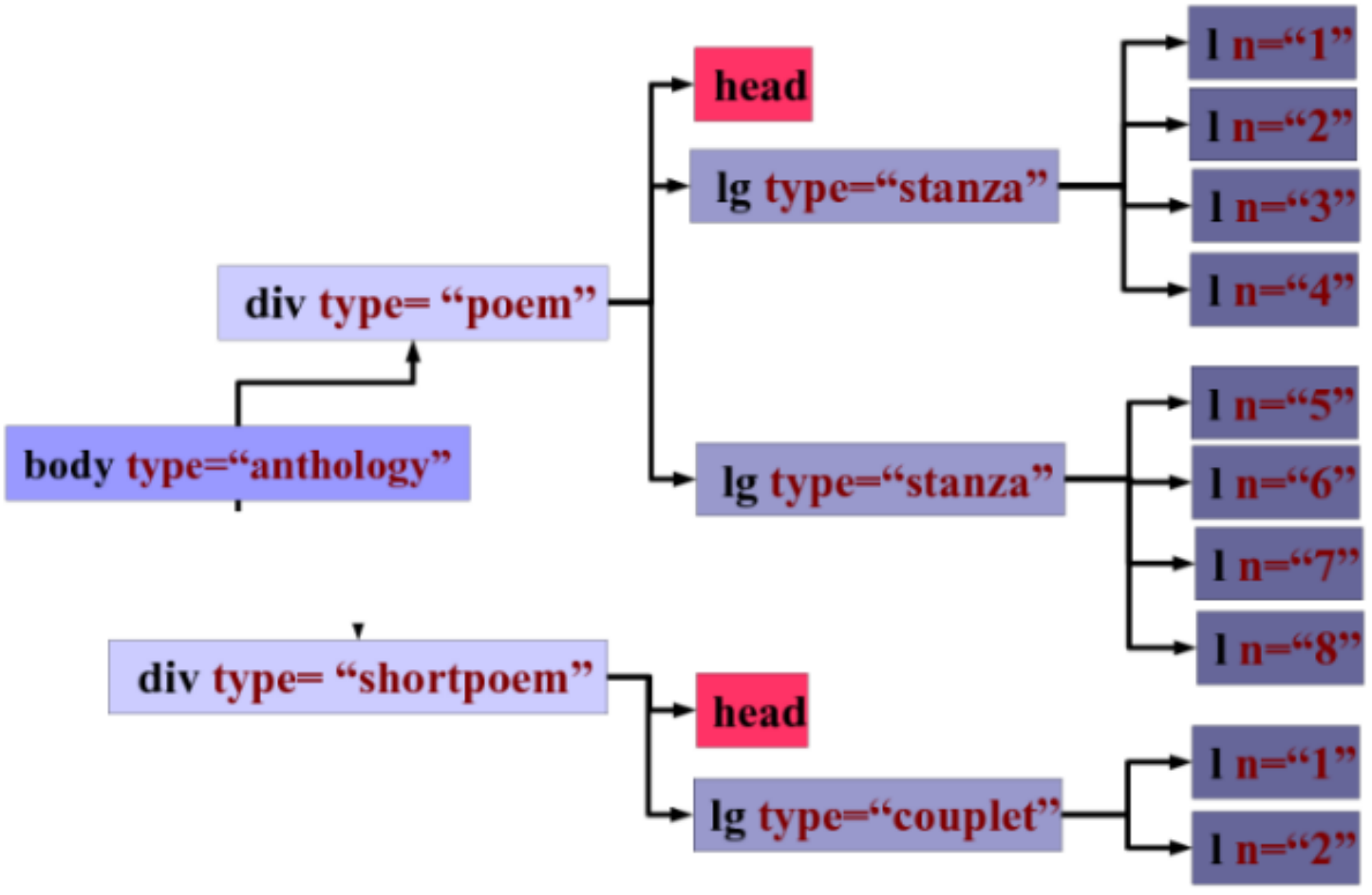
XPath : une feuille de route

Pour accéder aux composants d'un document XML, on spécifie un *chemin*, spécifiant les nœuds qu'il faut traverser pour arriver à la partie souhaité

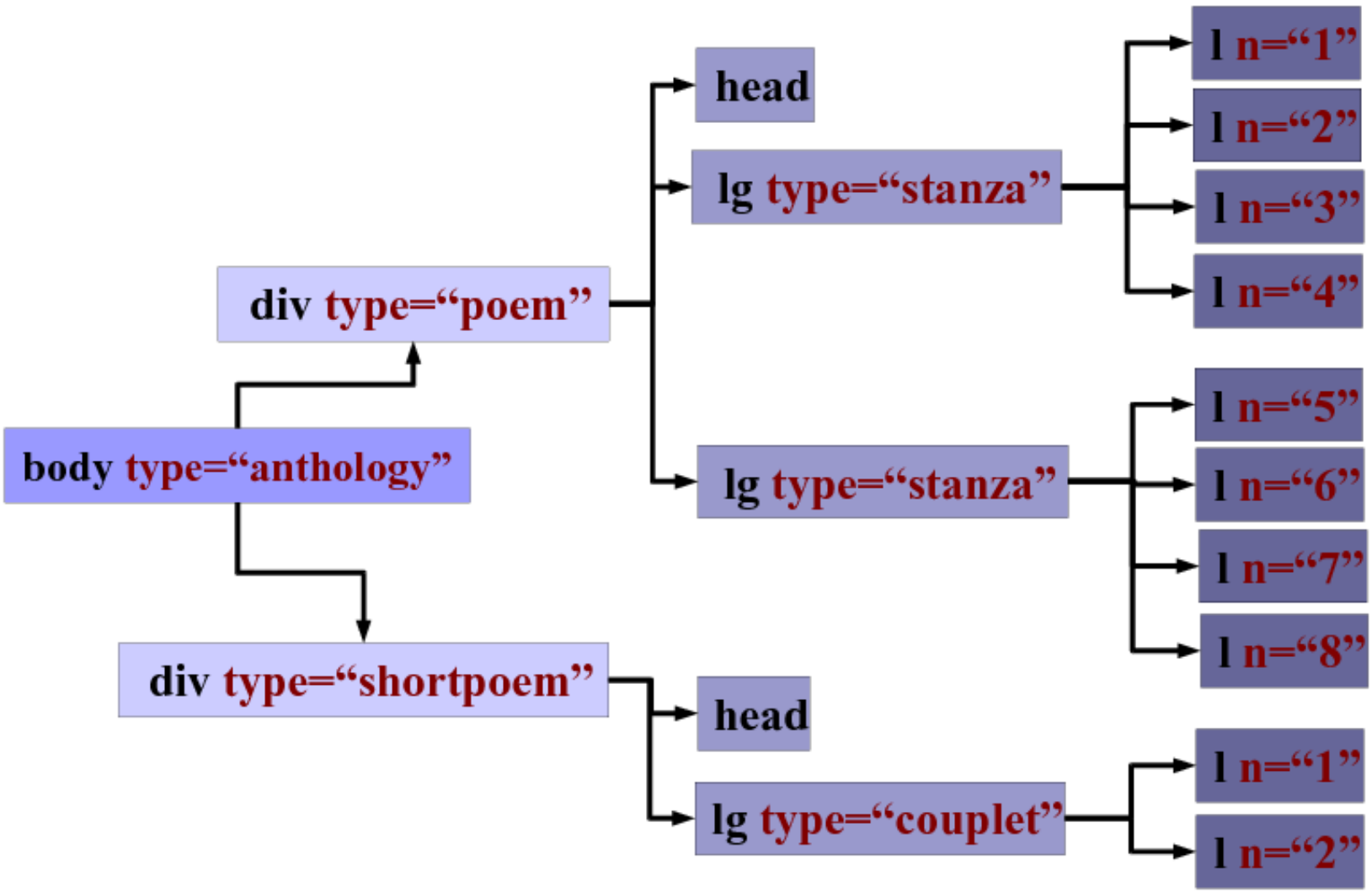
Par exemple, pour arriver aux <head>s dans cet exemple, on commence au <body>, puis passe à un fils, et ensuite on arrive à un <head>

En XPath, on dit : `/body/div/head`

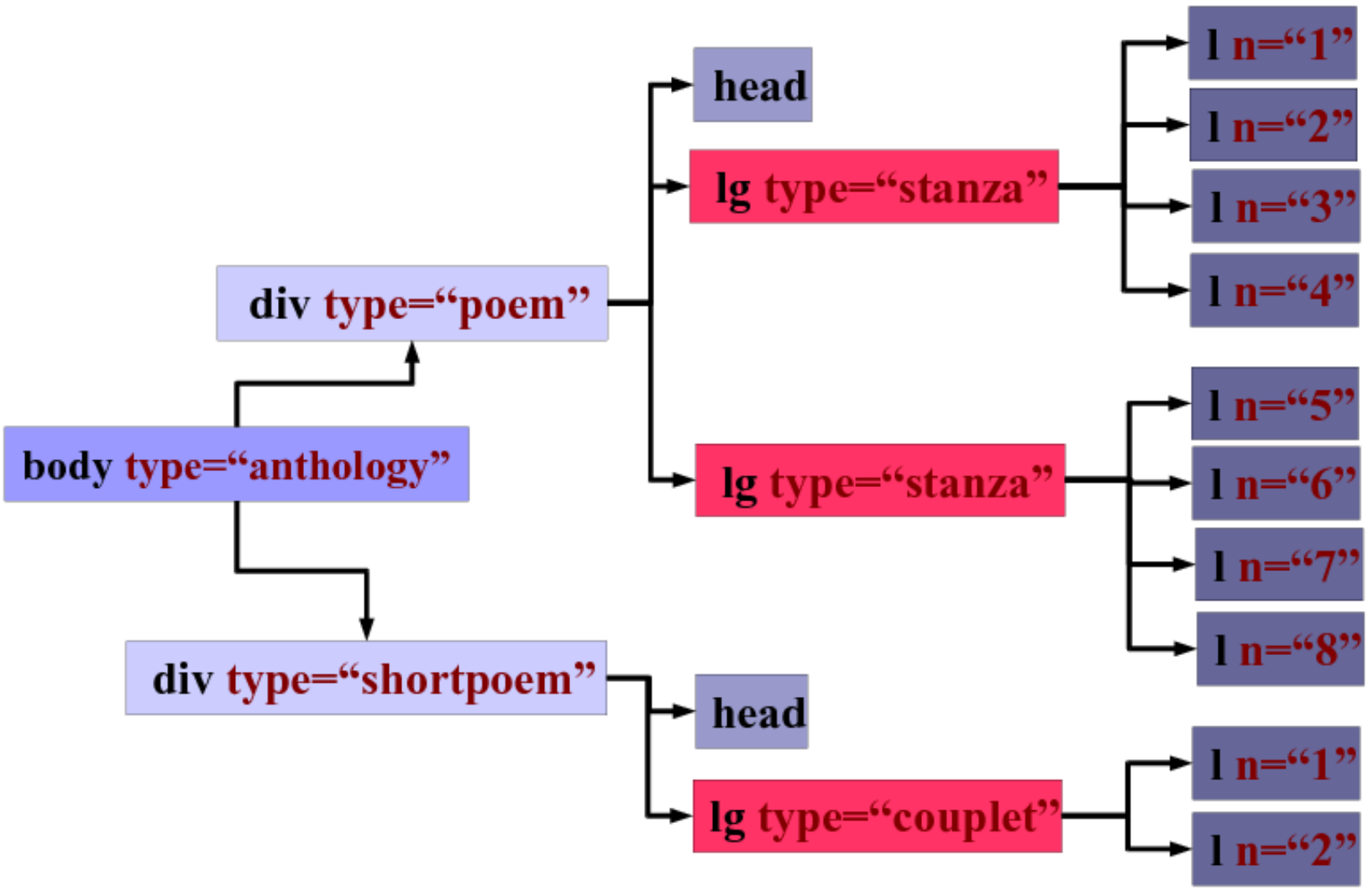
/body/div/head



/body/div/lg ?


body type="anthology"
div type="poem"
head
lg type="stanza"
l n="1"
l n="2"
l n="3"
l n="4"
lg type="stanza"
l n="5"
l n="6"
l n="7"
l n="8"
div type="shortpoem"
head
lg type="couplet"
l n="1"
l n="2"

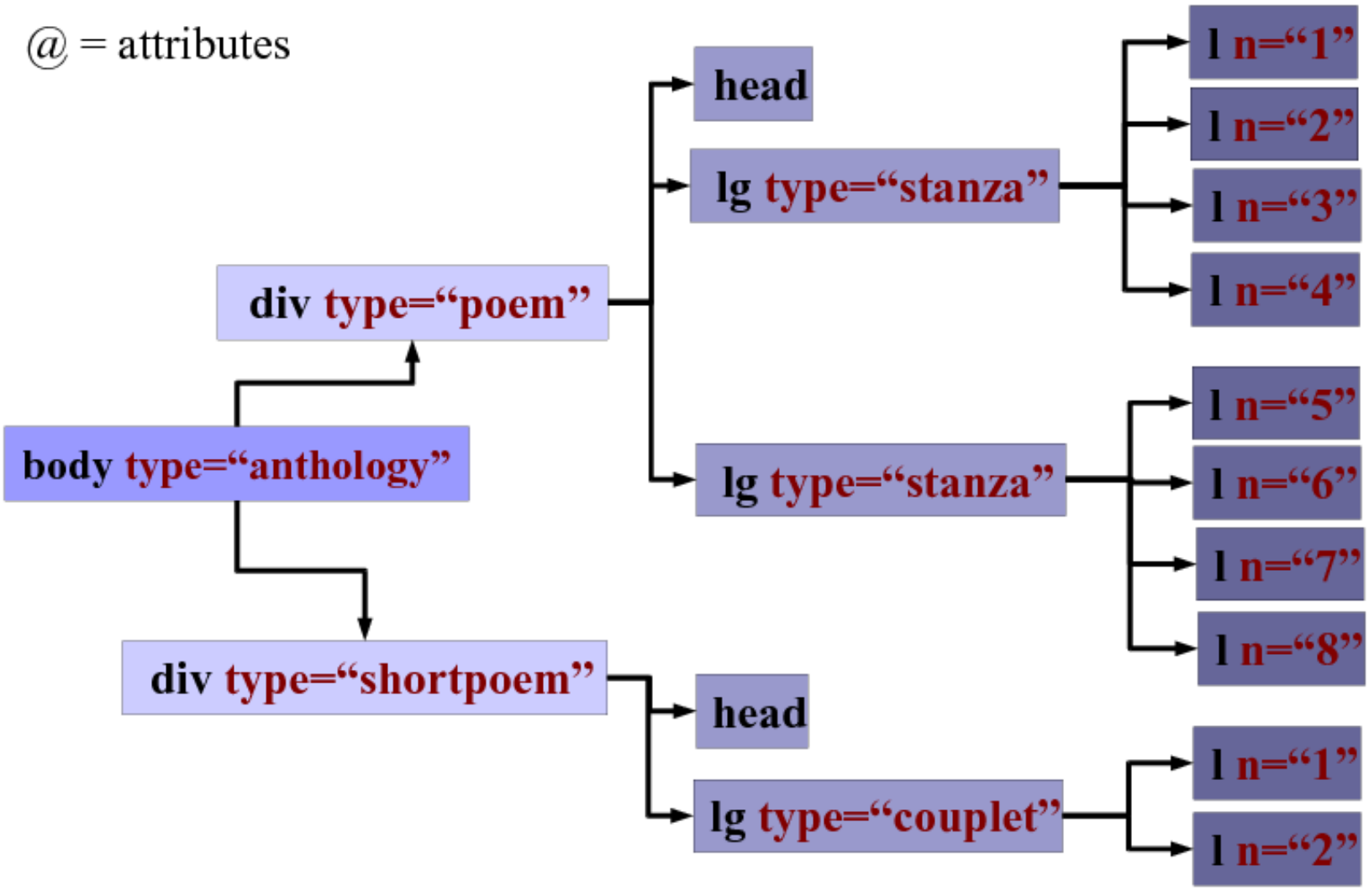
/body/div/lg


body type="anthology"
div type="poem"
head
lg type="stanza"
l n="1"
l n="2"
l n="3"
l n="4"
lg type="stanza"
l n="5"
l n="6"
l n="7"
l n="8"
div type="shortpoem"
head
lg type="couplet"
l n="1"
l n="2"

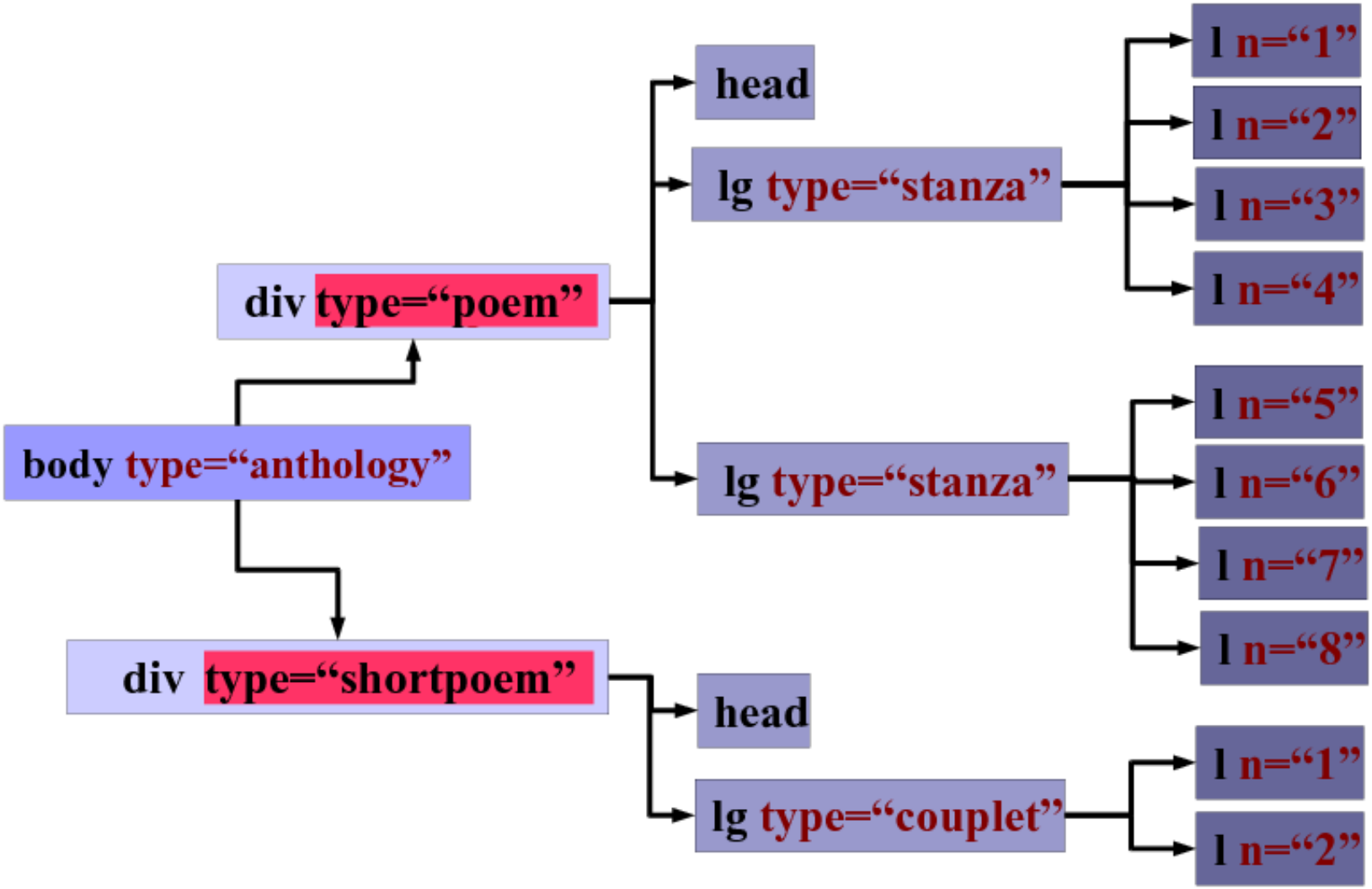
Les étapes

- Chaque étape sur ce chemin n'est pas forcément un élément XML...
- on peut aussi regarder les attributs
- ou des morceaux de texte

/body/div/@type ?



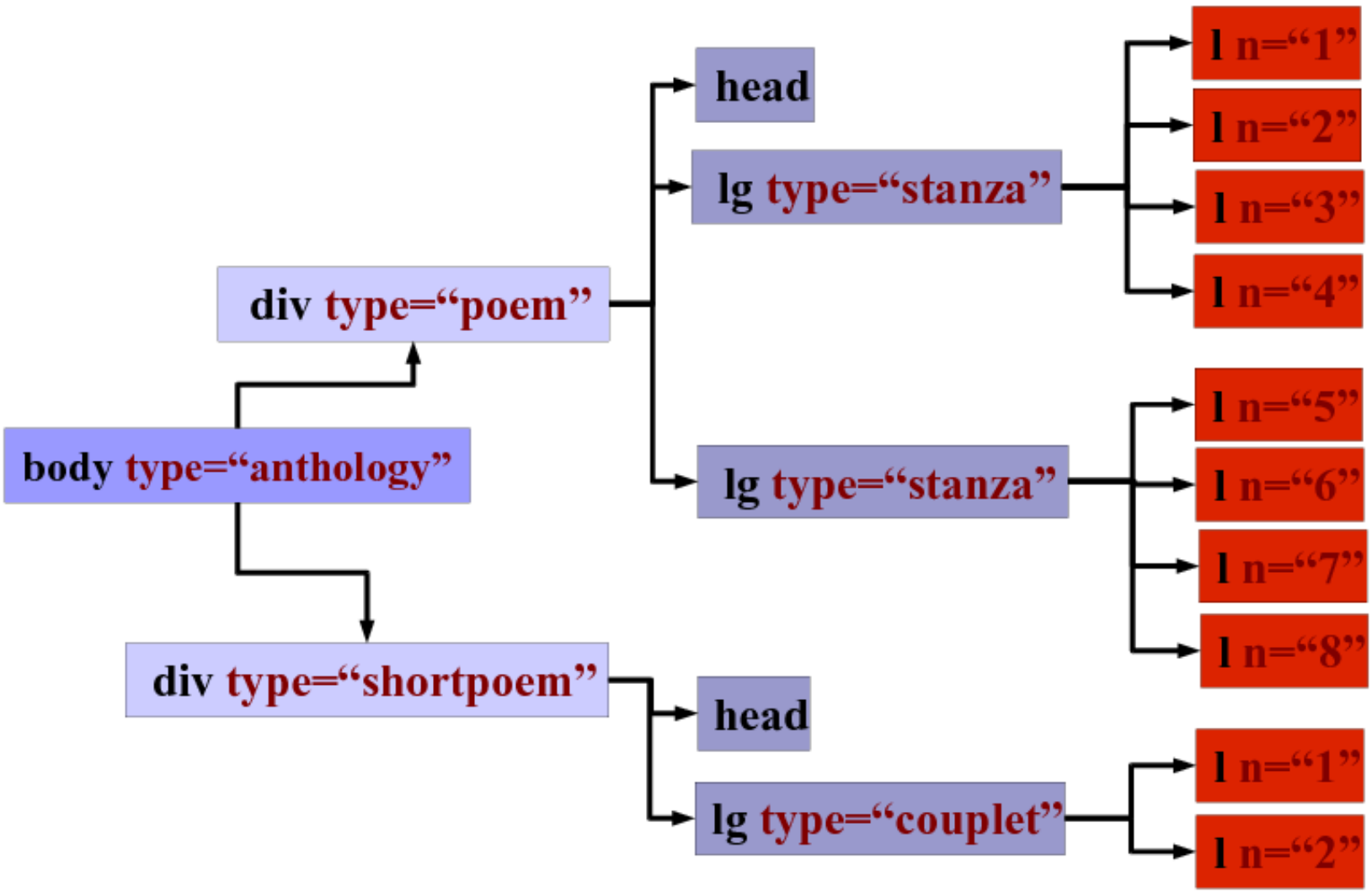
/body/div/@type



Les sélections

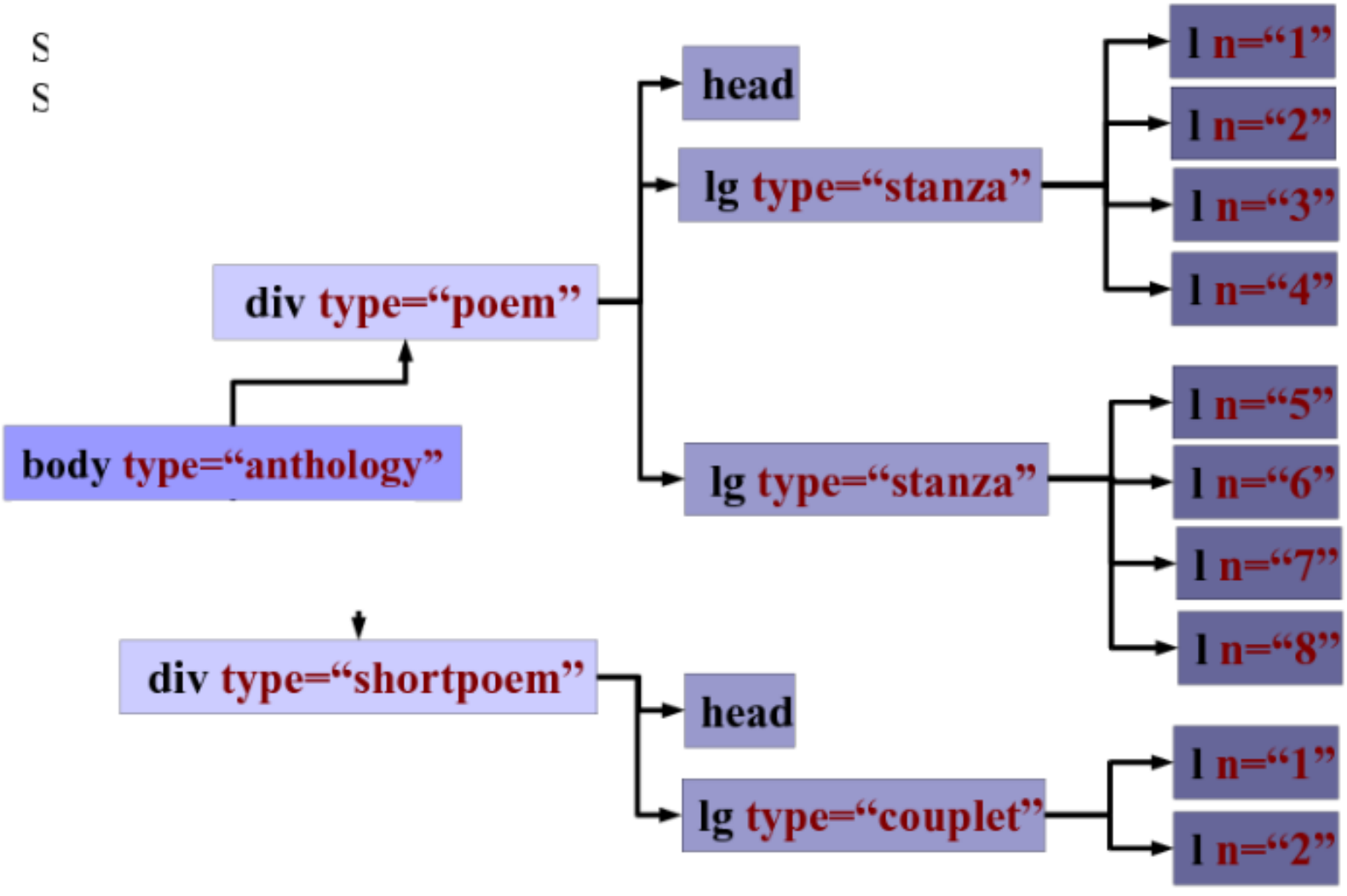
- On peut sélectionner parmi les nœuds résultants, en forme de restriction entre parenthèses [et]
- Une restriction peut tenir en compte la valeur d'un attribut
- ou la position ordinale du nœud dans l'arbre
- ou l'existence d' un élément du type indiqué

/body/div/lg/l

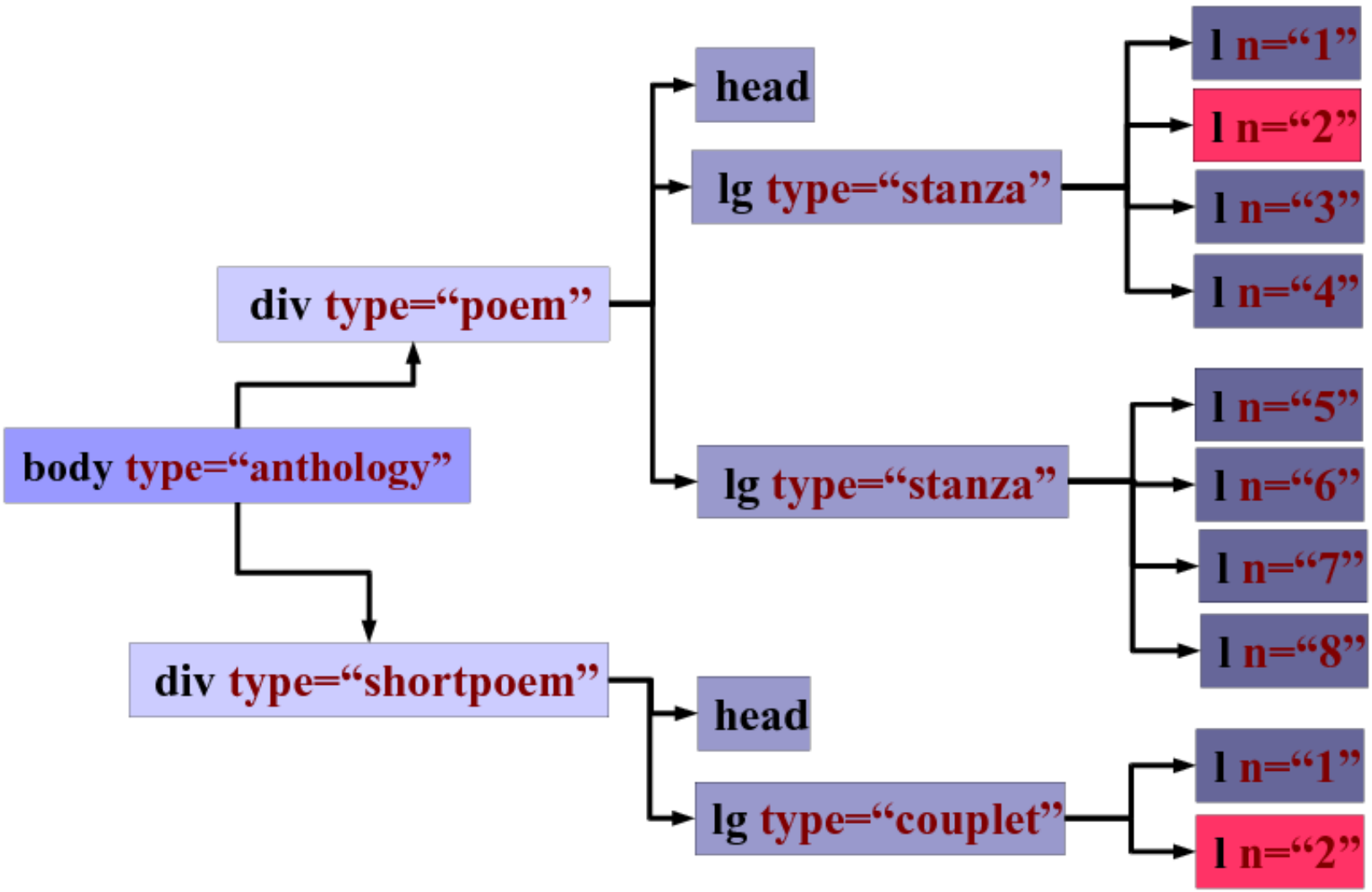

body type="anthology"
div type="poem"
head
lg type="stanza"
l n="1"
l n="2"
l n="3"
l n="4"
lg type="stanza"
l n="5"
l n="6"
l n="7"
l n="8"
div type="shortpoem"
head
lg type="couplet"
l n="1"
l n="2"

/body/div/lg/l[@n="2"] ?

S
S



/body/div/lg/l[@n="2"]



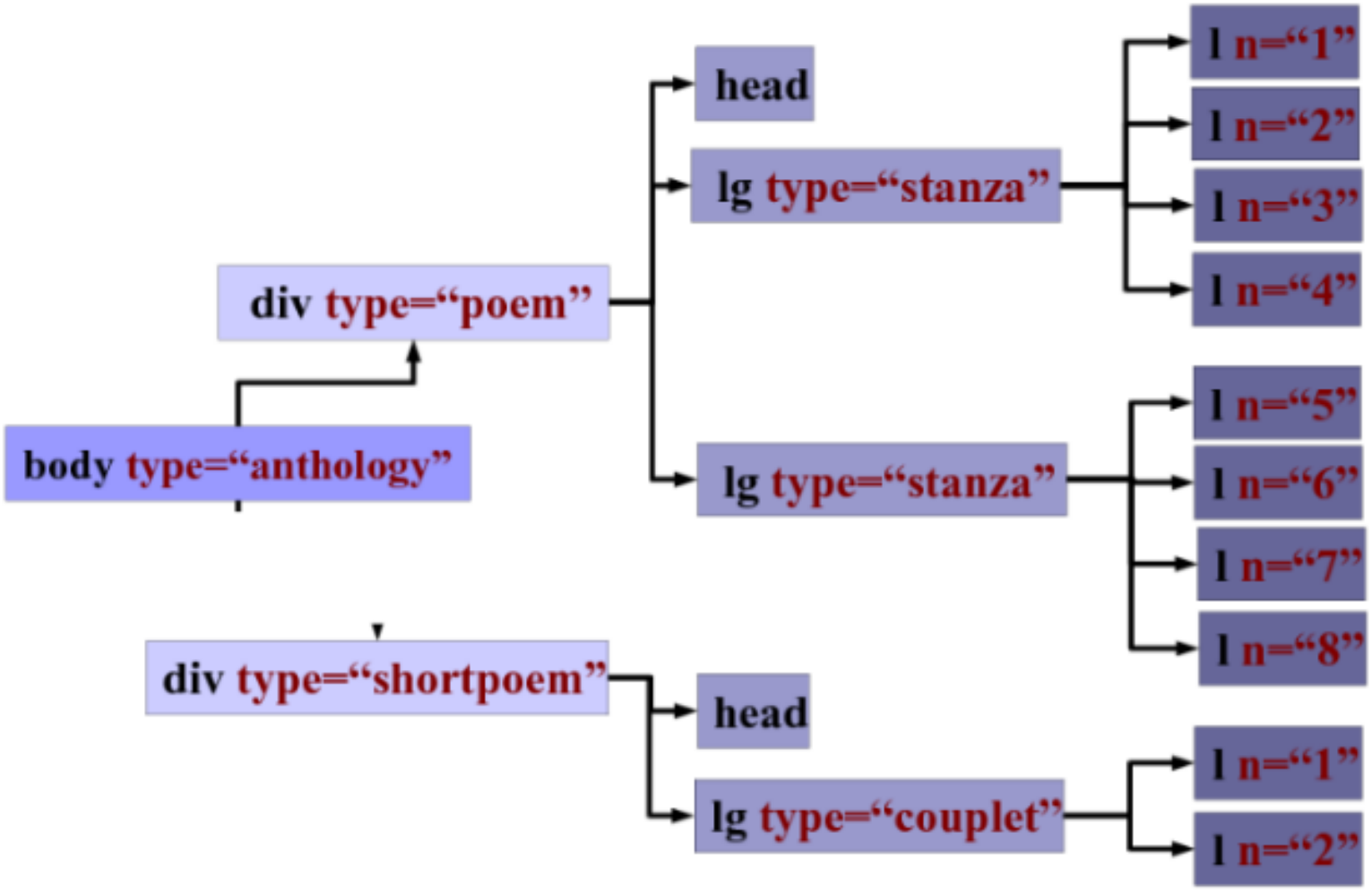
Le point de départ

On peut partir de n'importe quel point dans l'arborescence:

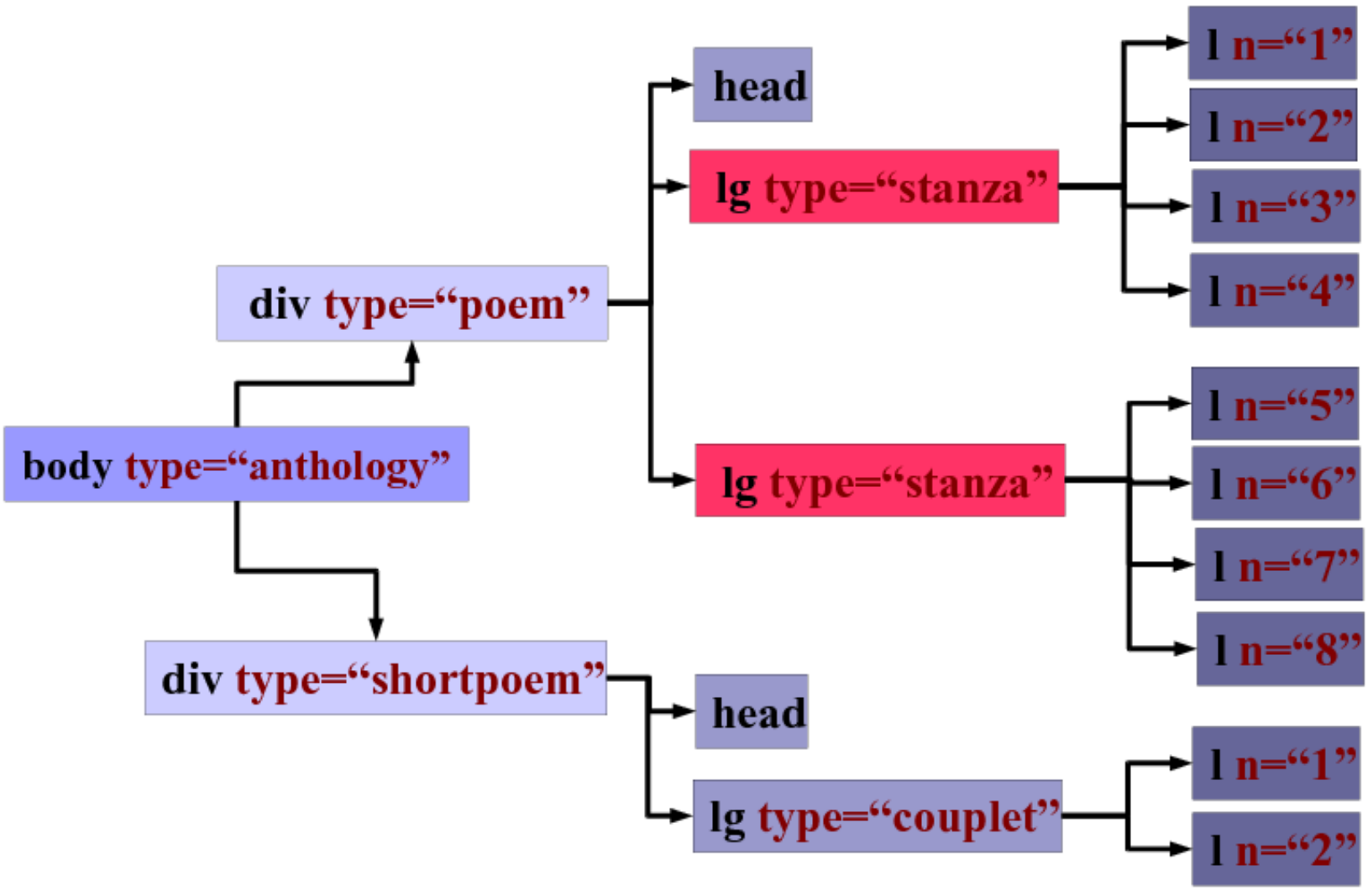
- `//` signifie 'n'importe où'
- `..` signifie 'mon parent'

On peut aussi naviguer dans la hiérarchie, en se servant des *axes* tels que `ancestor::`, `following-sibling::`, `descendant::` ...

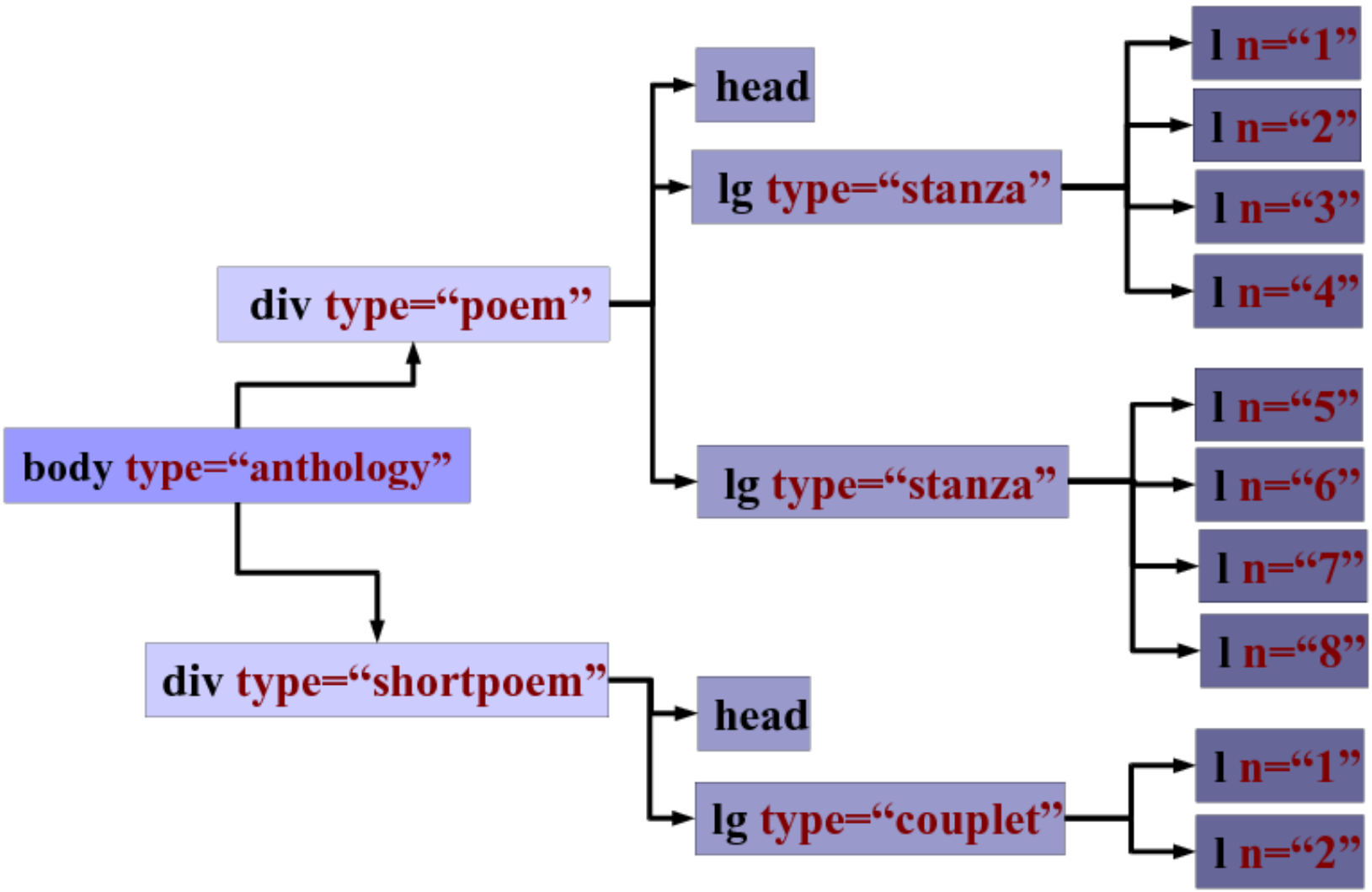
//lg[@type="stanza"] ?



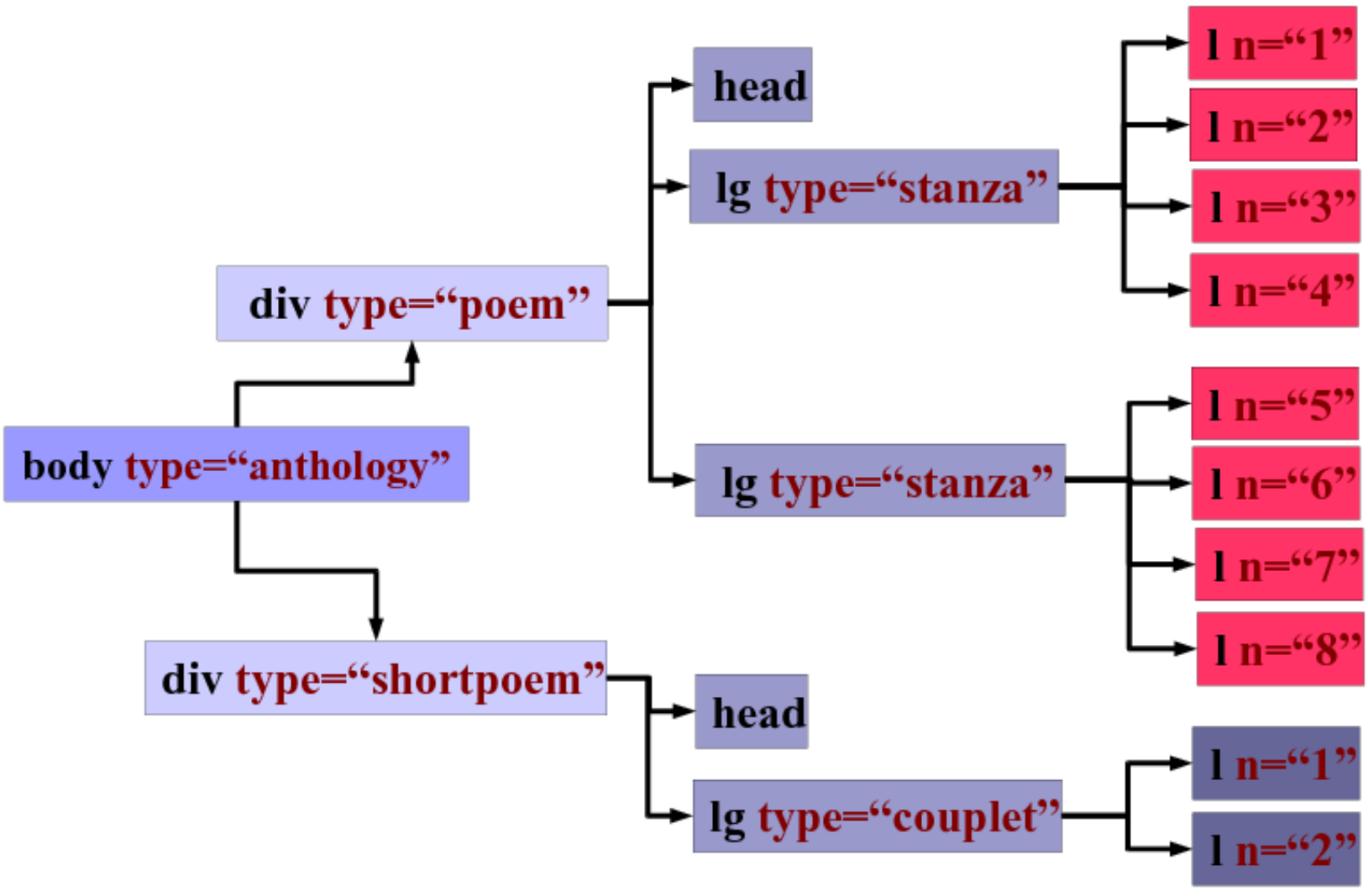
//lg[@type="stanza"]



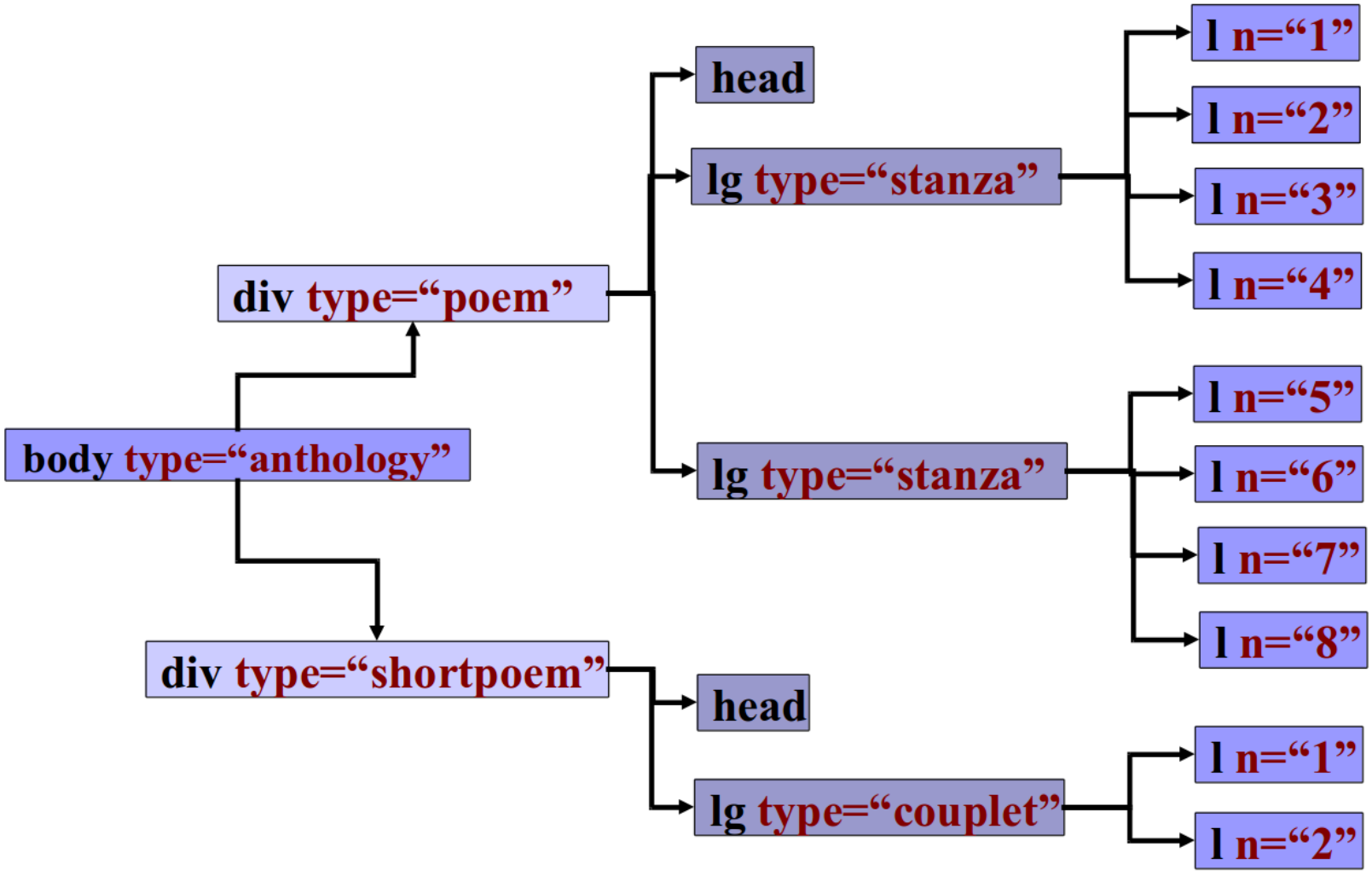
//div[@type="poem"]//l ?



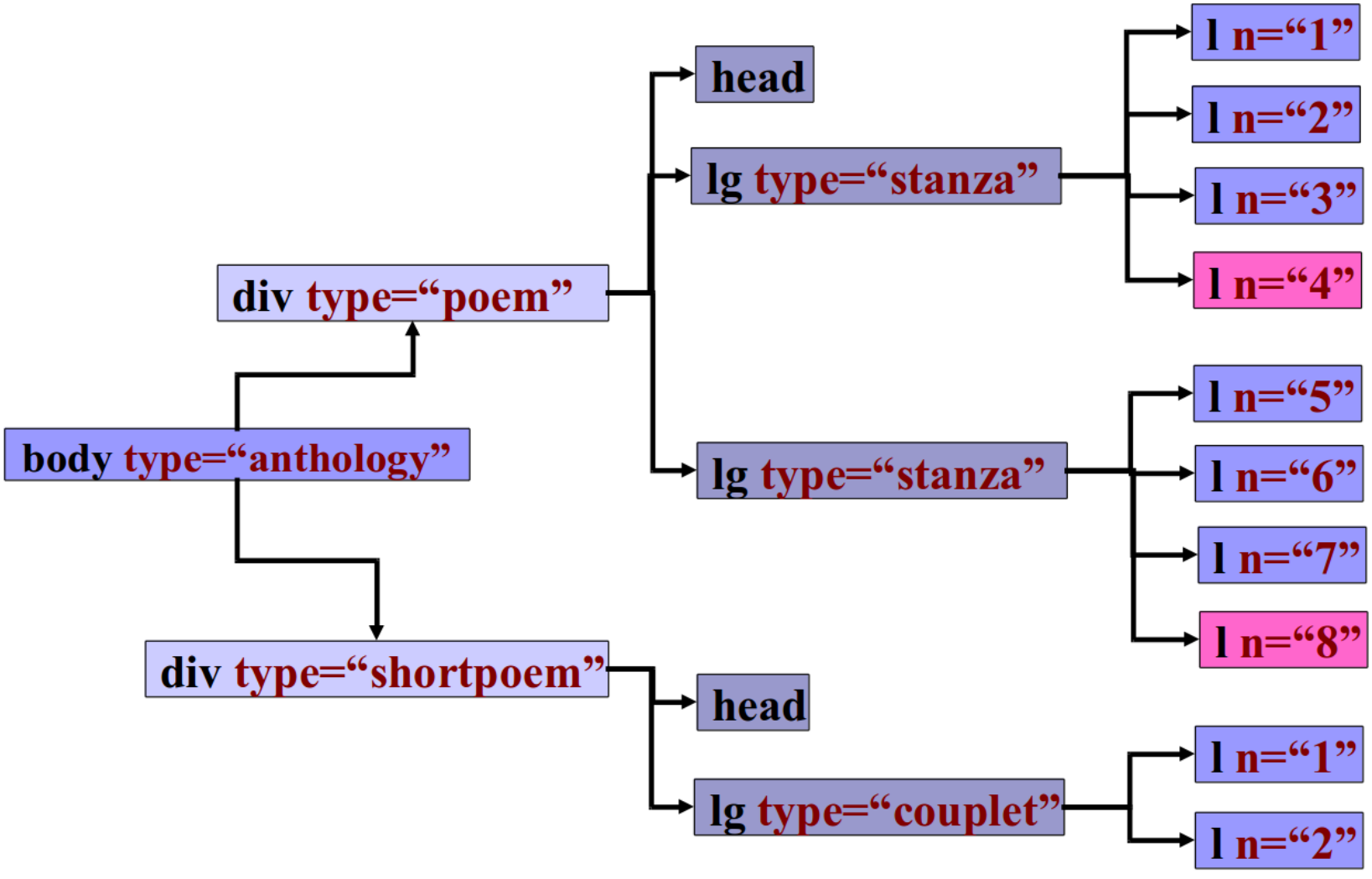
//div[@type="poem"]//l



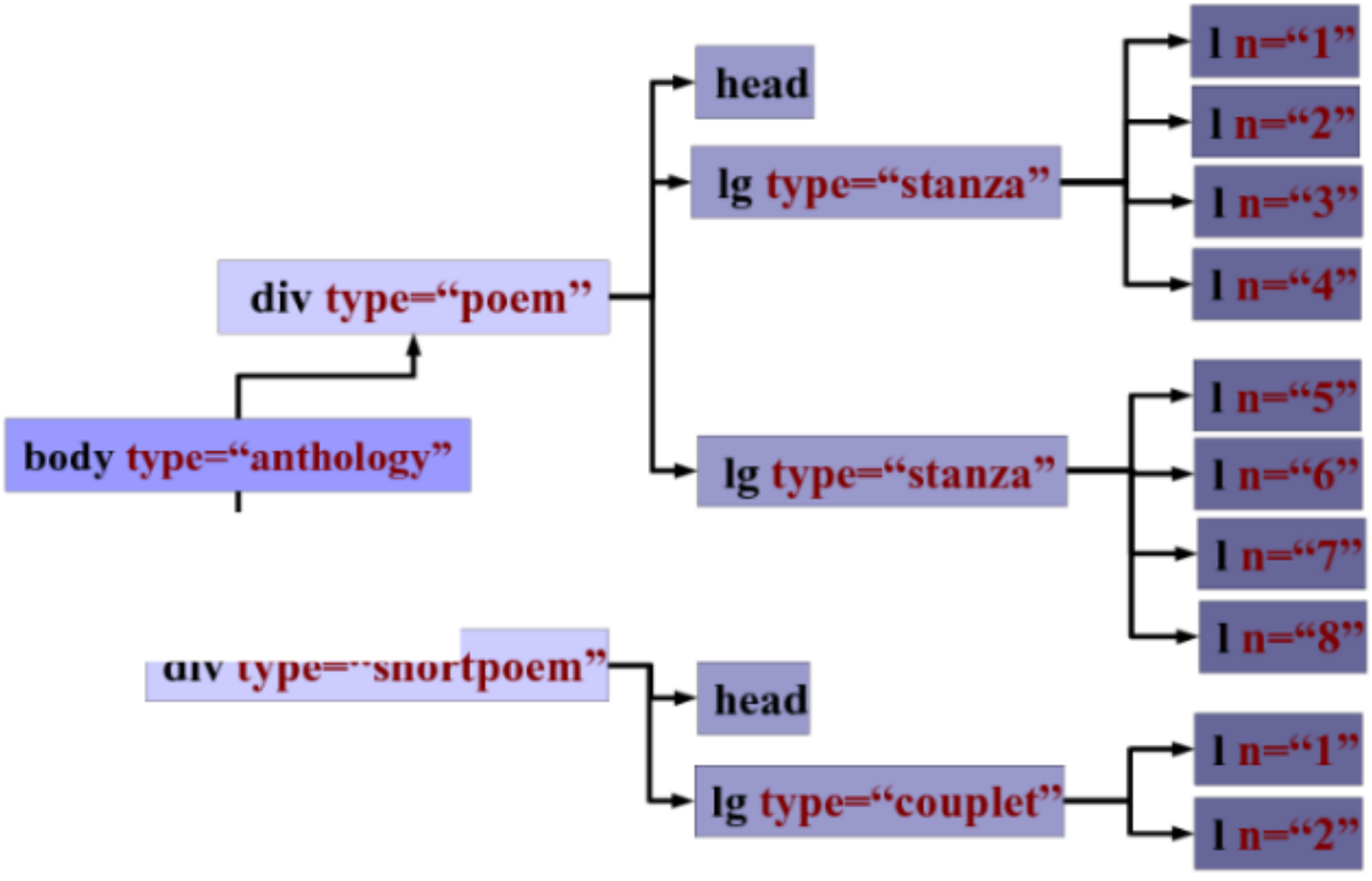
//l[4] ?


body type="anthology"
div type="poem"
head
lg type="stanza"
l n="1"
l n="2"
l n="3"
l n="4"
lg type="stanza"
l n="5"
l n="6"
l n="7"
l n="8"
div type="shortpoem"
head
lg type="couplet"
l n="1"
l n="2"

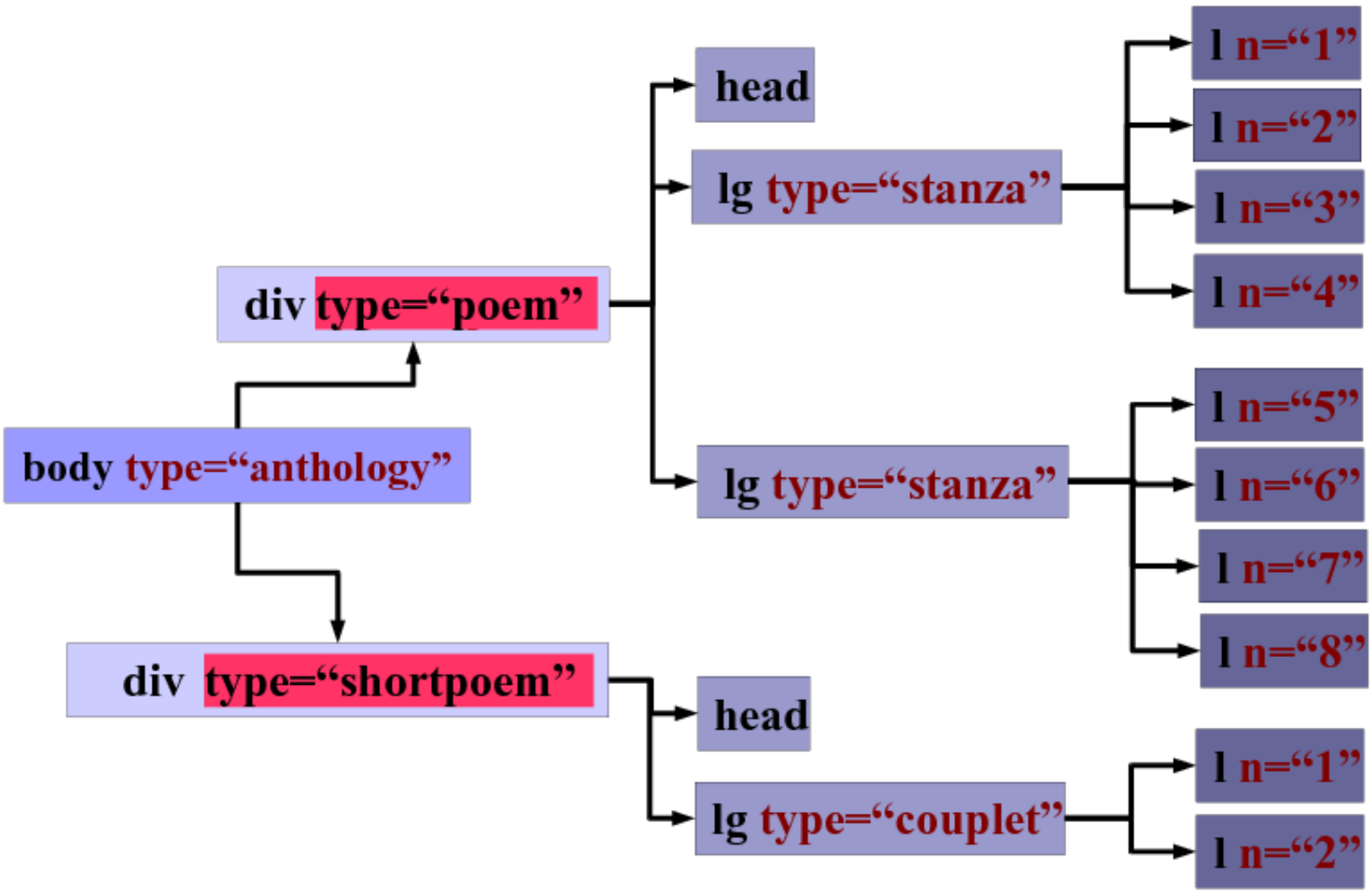
//l[4]


body type="anthology"
div type="poem"
head
lg type="stanza"
l n="1"
l n="2"
l n="3"
l n="4"
lg type="stanza"
l n="5"
l n="6"
l n="7"
l n="8"
div type="shortpoem"
head
lg type="couplet"
l n="1"
l n="2"

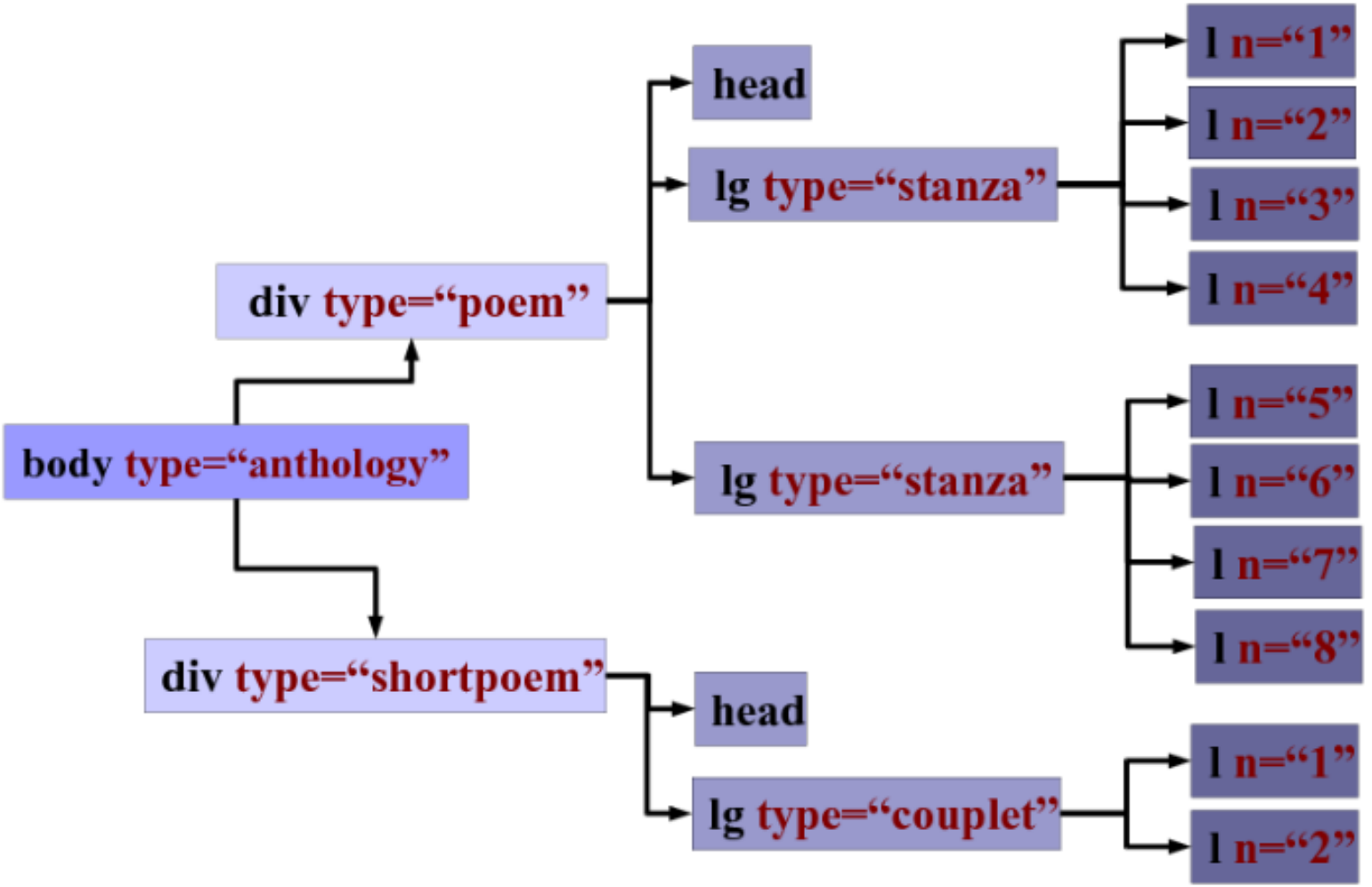
//lg/../@type ?


body type="anthology"
div type="poem"
head
lg type="stanza"
l n="1"
l n="2"
l n="3"
l n="4"
lg type="stanza"
l n="5"
l n="6"
l n="7"
l n="8"
div type="shortpoem"
head
lg type="couplet"
l n="1"
l n="2"

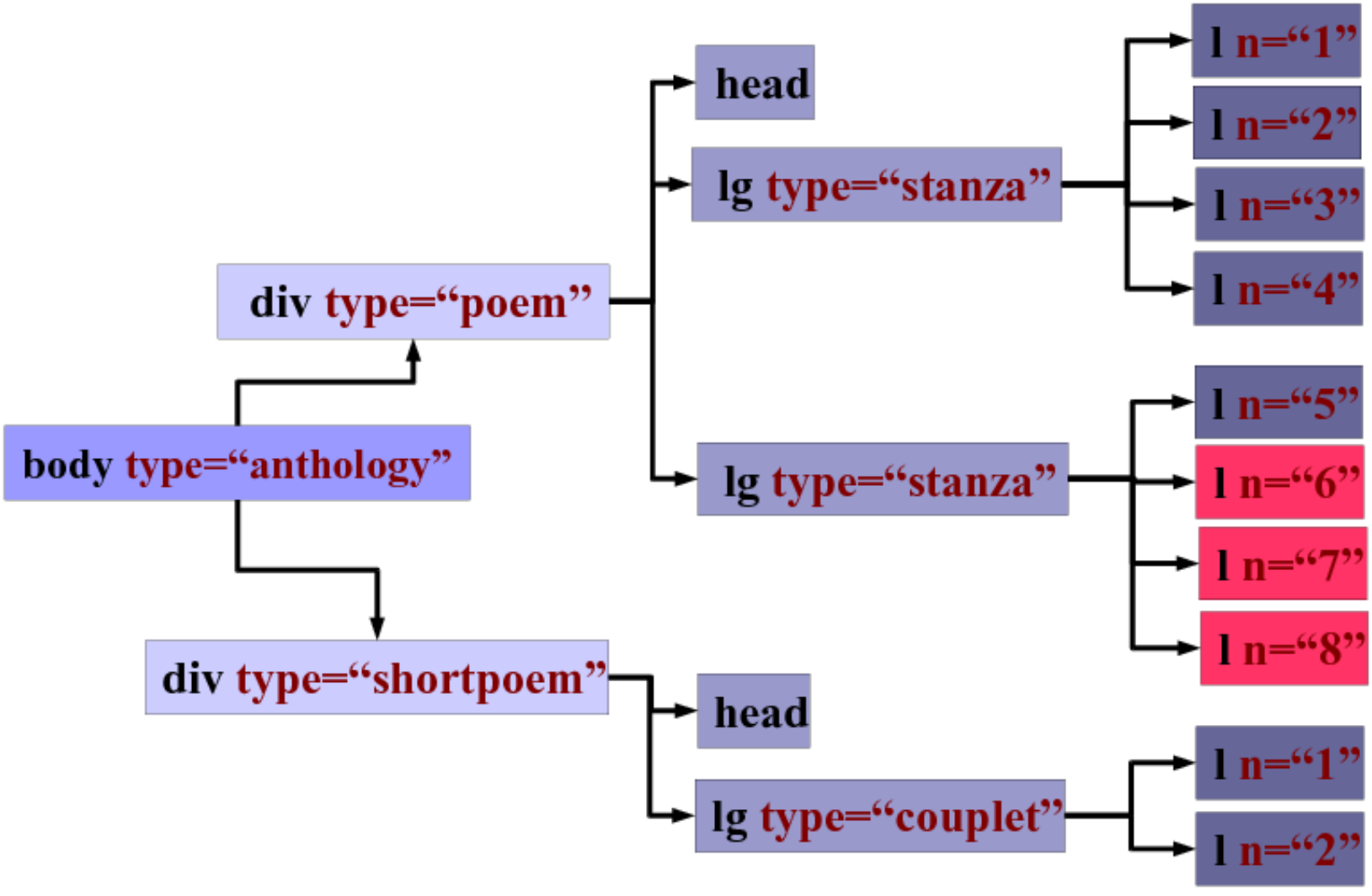
//lg/../@type


body type="anthology"
div type="poem"
head
lg type="stanza"
l n="1"
l n="2"
l n="3"
l n="4"
lg type="stanza"
l n="5"
l n="6"
l n="7"
l n="8"
div type="shortpoem"
head
lg type="couplet"
l n="1"
l n="2"

//l[@n > 5] ?


body type="anthology"
div type="poem"
head
lg type="stanza"
l n="1"
l n="2"
l n="3"
l n="4"
lg type="stanza"
l n="5"
l n="6"
l n="7"
l n="8"
div type="shortpoem"
head
lg type="couplet"
l n="1"
l n="2"

//l[@n > 5]


body type="anthology"
div type="poem"
head
lg type="stanza"
l n="1"
l n="2"
l n="3"
l n="4"
lg type="stanza"
l n="5"
l n="6"
l n="7"
l n="8"
div type="shortpoem"
head
lg type="couplet"
l n="1"
l n="2"

Fonctions XPath

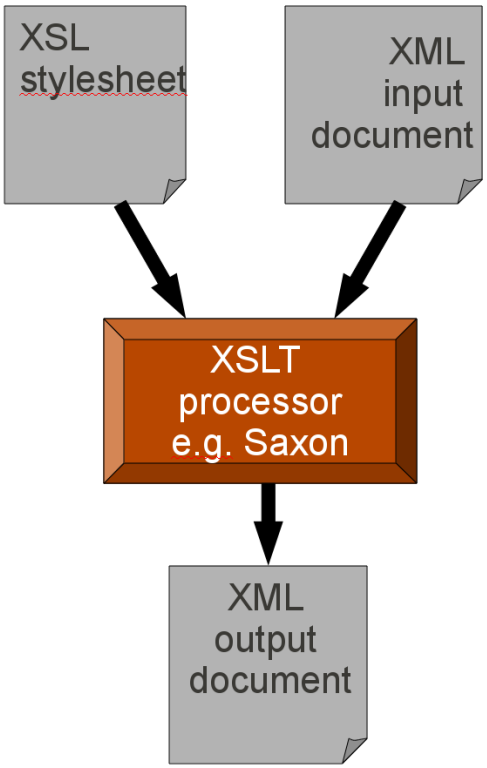
XPath fournit également une librairie extensive de fonctions utiles. On signale ici seulement quelques-unes :

- `count(x)` fournit le nombre des nœuds dans l'arbre x
- `position()` fournit le nombre ordinal du nœud courant par rapport à son contexte
- `last()` fournit le nombre ordinal du dernier nœud courant par rapport à son contexte
- `contains(x,y)` test l'existence de la chaîne y dans le morceau de texte x

Premier exercice

Tester votre compréhension de XPath avec la première partie de l'exercice...

Comment se servir d' XSLT ?



XSLT est un langage de transformation

Une transformation simplissime>

A partir de ceci :

```
<ref target="http://www.tei-c.org">The TEI website</ref>
```

on veut produire :

```
<a href="http://www.tei-c.org">The TEI website</a>
```

donc on va ....

- transformer l'élément <ref> de la TEI dans un élément <a> de (x)HTML
- transformer son attribut *@target* dans un attribut *@href*

Comment exprimer cela en XSL?

```
<xsl:stylesheet xpath-default-namespace="http://www.tei-
c.org/ns/1.0"
 version="2.0">
 <xsl:template match="ref">
  <a href="{@target}">
   <xsl:apply-templates/>
  </a>
 </xsl:template>
</xsl:stylesheet>
```

Une transformation plus typique

A partir de ceci :

```
<div type="recette" n="34">
 <head>Pasta pour les débutants</head>
 <list>
  <item>pates</item>
  <item>fromage râpé</item>
 </list>
 <p>Faire bouiller les pates, et mélanger avec le
fromage.</p>
</div>
```

on veut produire :

```
<html>
 <h1>34: Pasta pour les novices</h1>
 <p>Ingrédients: pates fromage râpé</p>
 <p>Faire bouiller les pates, et melanger avec le
fromage.</p>
</html>
```

Comment exprimer cela en XSL?

```
<xsl:stylesheet xpath-default-namespace="http://www.tei-
c.org/ns/1.0"
 version="2.0">
 <xsl:template match="div">
  <html>
   <h1>
    <xsl:value-of select="@n"/>:
   <xsl:value-of select="head"/>
   </h1>
   <p>Ingrédients:
   <xsl:apply-templates select="list/item"/>
   </p>
   <p>
    <xsl:value-of select="p"/>
   </p>
  </html>
 </xsl:template>
</xsl:stylesheet>
```

Une feuille de style XSLT

- est un document XML, contenant des éléments de l'espace de noms `http://www.w3.org/1999/XSL/Transform`
- <xsl:stylesheet> (élément racine de tout stylesheet) permet de spécifier tous les noms d'espace utilisés, un nom d'espace par défaut, et la version du norme XSLT employé (1 ou 2)
- <xsl:output> : spécifie quelques options pour l'arbre de sortie, par exemple son format (HTML, XML, TEXT...), encodage (ISO-8859-1, UTF-8 ...) etc.

```
<xsl:stylesheet xpath-default-namespace="http://www.tei-
c.org/ns/1.0"
 version="2.0">
 <xsl:output method="html"
  encoding="ISO-8859-1"/>
 <xsl:template match="/">
  <xsl:apply-templates/>
 </xsl:template>
</xsl:stylesheet>
```

Dix éléments XSLT essentiels

- <xsl:template> spécifie un modèle de transformation
- <xsl:apply-templates> applique des templates
- <xsl:value-of> sort une valeur
- <xsl:text> sort un morceau de texte
- <xsl:element>, <xsl:attribute> et <xsl:comment> sortent un élément, attribut, ou commentaire
- <xsl:if> et <xsl:choose> actions conditionnels
- <xsl:for-each> bouclage des actions
- <xsl:variable> définition de variable
- <xsl:number> effectue une numerotation
- <xsl:sort> effectue un tri

<xml:template>

Cet élément spécifie un modèle (des actions) à appliquer à l'arborescence spécifiée par son attribut *@match*

Il peut contenir d'autres éléments XSL, des éléments d'autres noms d'espace (qui seront copiés), ou rien de tout.

```
<xsl:stylesheet xpath-default-namespace="http://www.tei-
c.org/ns/1.0"
 version="2.0">
 <xsl:template match="div">
<!-- .... actions pour les éléments div... -->
 </xsl:template>
 <xsl:template match="head">
<!-- .... actions pour tous les éléments head... -->
 </xsl:template>
 <xsl:template match="div/head">
<!-- .... actions pour les éléments head contenus par un
div....-->
 </xsl:template>
 <xsl:template match="teiHeader"/>
</xsl:stylesheet>
```

Les règles d'or de XSLT

Par défaut, le document est a traiter élément par élément...

1. Si aucun template ne correspond à un élément, traiter les éléments qu'il contient
2. Si aucun élément reste à traiter par regle 1, sortir les morceaux de texte contenus par l' élément
3. Un élément n'est traite que si un template lui correspond
4. L'ordre des templates dans le stylesheet est sans significance
5. Tout partie du document est traitable part tout template, éventuellement plusieurs fois
6. Un stylesheet ne peut contenir que de XML bien-forme

Contenu d'un template

Les éléments XML d'un nom d'espaces autre que le XSL se trouvant dans un template sont sortis sans changement.

Les fragments textuels (plus ou moins) pareils.

Un template vide requiert la sortie de ... rien, donc (s'il est invoqué) il supprime les noeuds concernes.

Plusieurs templates peuvent être spécifies pour un même élément en des contextes divers

Comparer

```
<xsl:template match="head">
<!-- ... -->
</xsl:template>
```

avec

```
<xsl:template match="div/head">
<!-- ... -->
</xsl:template>
<xsl:template match="figure/head">
<!-- ... -->
</xsl:template>
```

En cas de conflit, c'est le template le plus spécifique qui gagne.

<xsl:apply-templates>

Cet élément rend disponible dans le contexte courant les règles contenus par les templates indiques dans son attribut *@select*. Si aucun template n'est indiqué, tous les templates sont disponibles.

```
<xsl:template match="/">
 <html>
  <xsl:apply-templates/>
 </html>
</xsl:template>
```

```
<xsl:template match="TEI">
 <xsl:apply-templates select="text"/>
</xsl:template>
```

Il est très utile pour varier l'ordre des sorties:

```
<xsl:template match="text">
 <h1>Corps du texte</h1>
 <xsl:apply-templates select="body"/>
 <h1>Pièces liminaires</h1>
 <xsl:apply-templates select="front"/>
 <xsl:apply-templates select="back"/>
</xsl:template>
```

<xsl:value-of>

Cet élément fait sortir la valeur d'un élément ou d'un attribut :

```
<xsl:value-
of select="/TEI/teiHeader/fileDesc/titleStmt/title"/>
```

Attention aux doublons potentiels!

```
<xsl:template match="div">
 <h2>
  <xsl:value-of select="@n"/>
  <xsl:value-of select="head"/>
 </h2>
 <xsl:apply-templates/>
</xsl:template>
<xsl:template match="div/head"/>
```

Pour en savoir plus

- A http://www.gchagnon.fr/cours/xml/ vous trouverez deux cours complets et très clairs
- Un texte classique : Philippe Rigaux et Bernd Amann *Comprendre XSLT*. O'Reilly, 2002.
- Beaucoup, beaucoup, d'autres ressources anglophones...