XSLT for Idiots

Lou Burnard Consulting

august 2016

Objectives

This is not a complete XSLT Training Course! Its purpose is just to ...

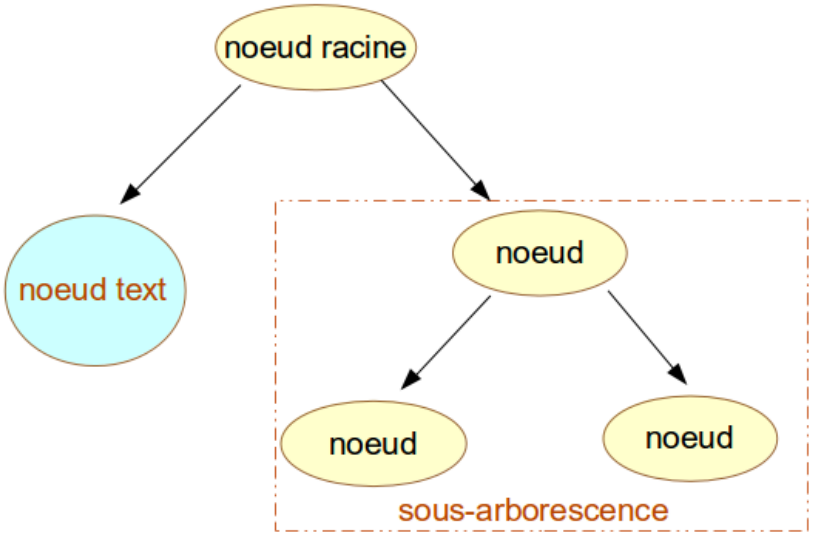
- give you a taste of what XSLT and XPath can do
- particularly when processing TEI-XML documents
- particularly TEI documents from the Humanities
- introduce a few of the essential concepts of XSLT

XSL: a set of complementary standards

- XPath: a standard syntax for addressing and accessing parts of an XML tree
- XSLT: a standard language for transforming XML trees
- XSL FO: an XML vocabulary for the description of page layout

Like XML itself, all three are developed and maintained by the W3C.

What is an *XML tree*?



- a set of *nodes*, organised hierarchically
- each node either has a *generic identifier* (its "type") or is a *text node*
- a single *root node* contains (or dominates) all the others
- each node can contain (or dominate)
  - a subtree
  - or a text node

In an XML Tree...

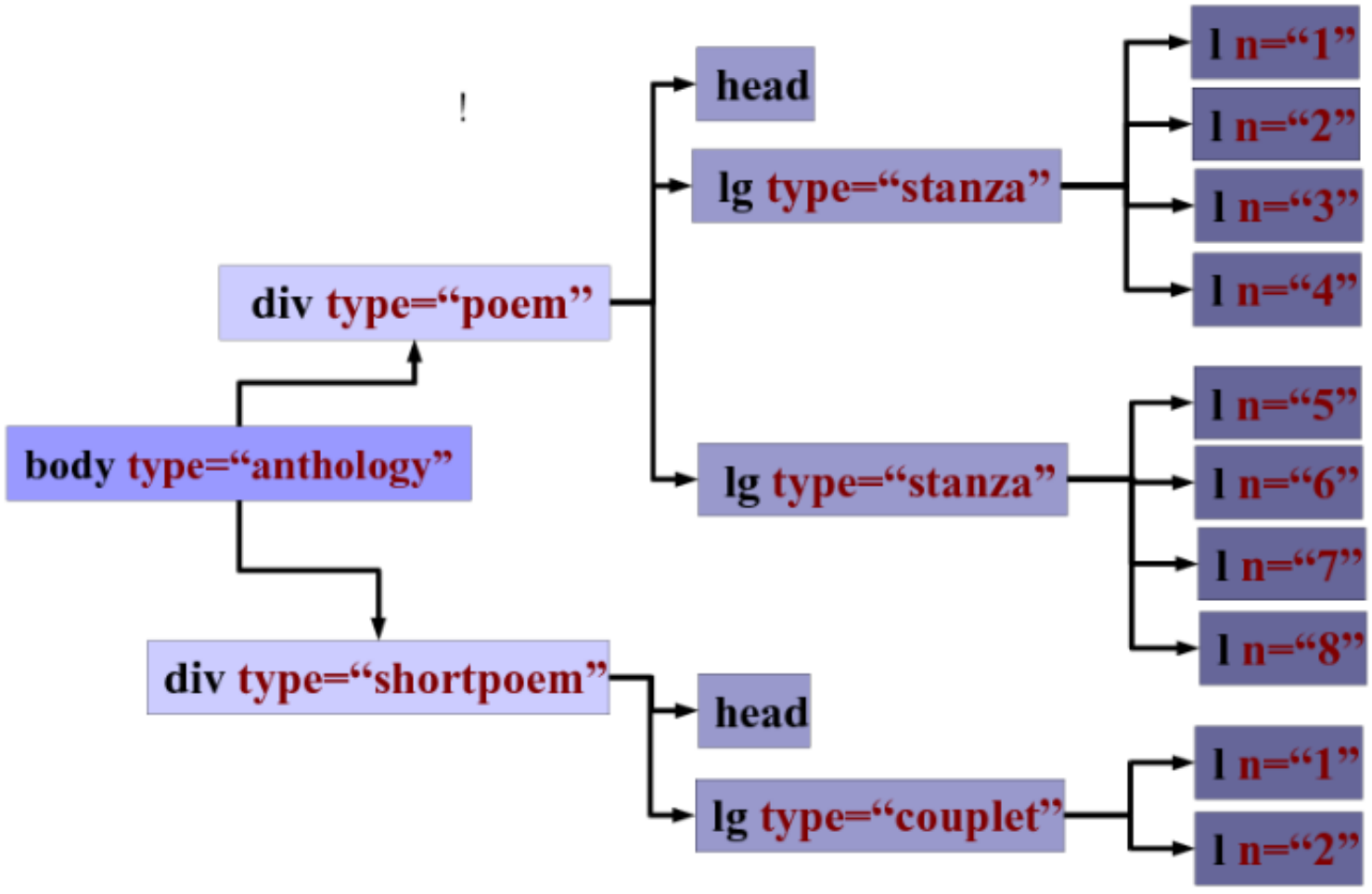
- each node corresponds with a named element
- the *attributes*of an element make up a sub-tree associated with a particular node
- each attribute has a *name* and a *value*

For example :

```
<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>
```

.. or, represented as a tree :

Un arborescence XML



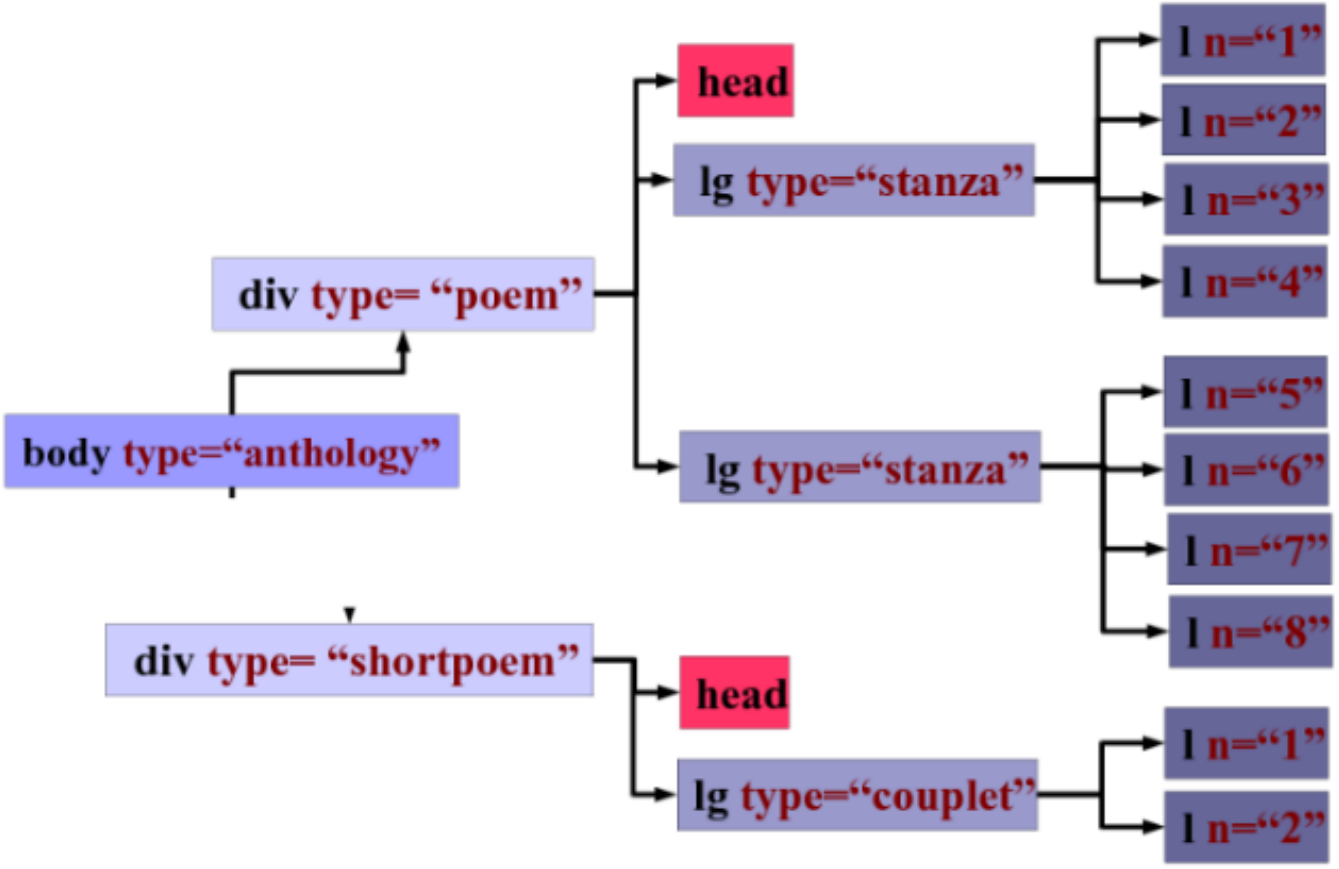
XPath : a road map

To access the components of an XML document, you supply a *path*, specifying the nodes you must pass through to get to the part you want

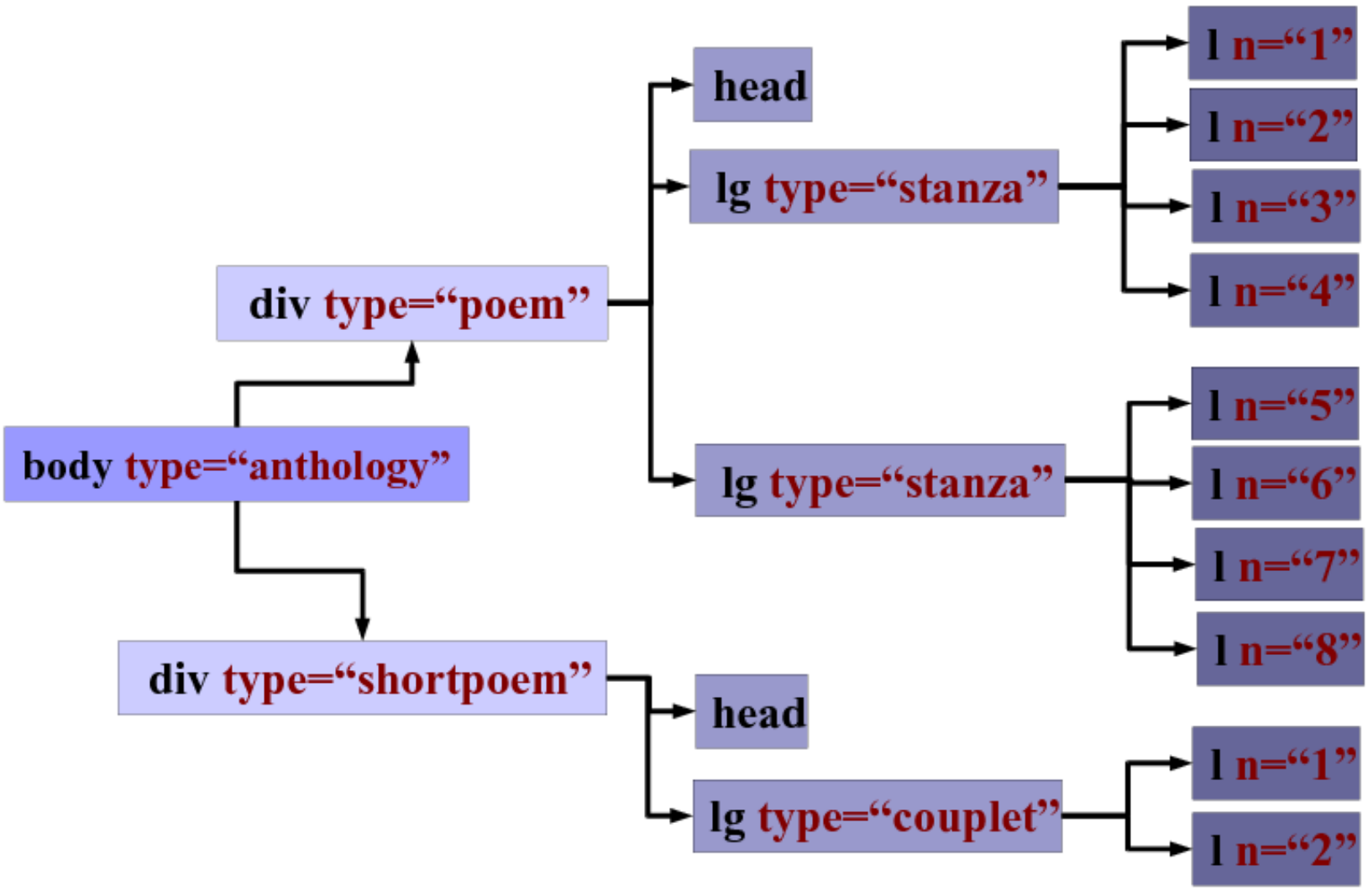
For example, to get to the <head>s in this example, you start at the <body>, then go down one level to a child , within which you go down a third level to find a <head>

In XPath, we say `body/div/head`

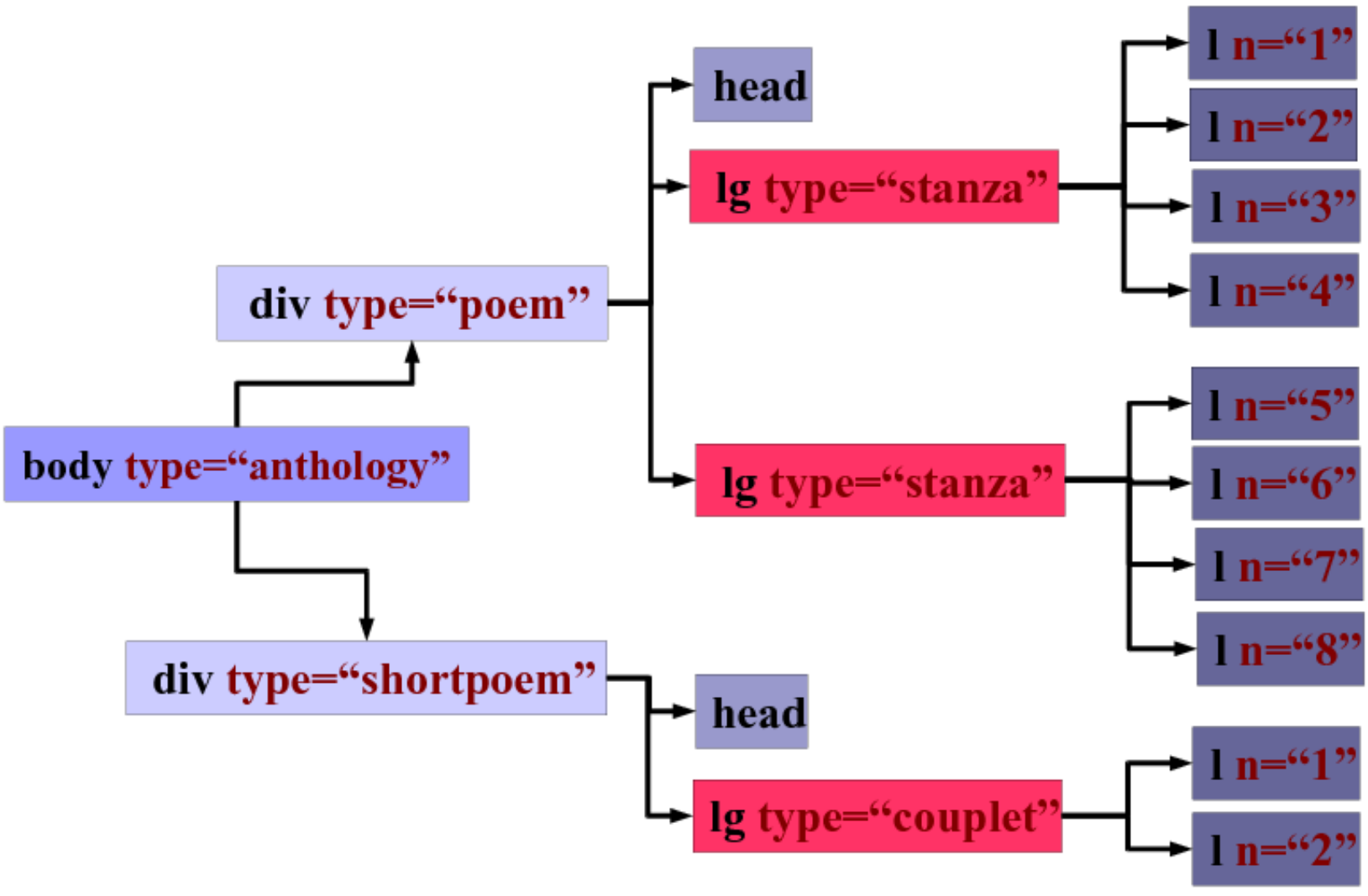
/body/div/head



/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"

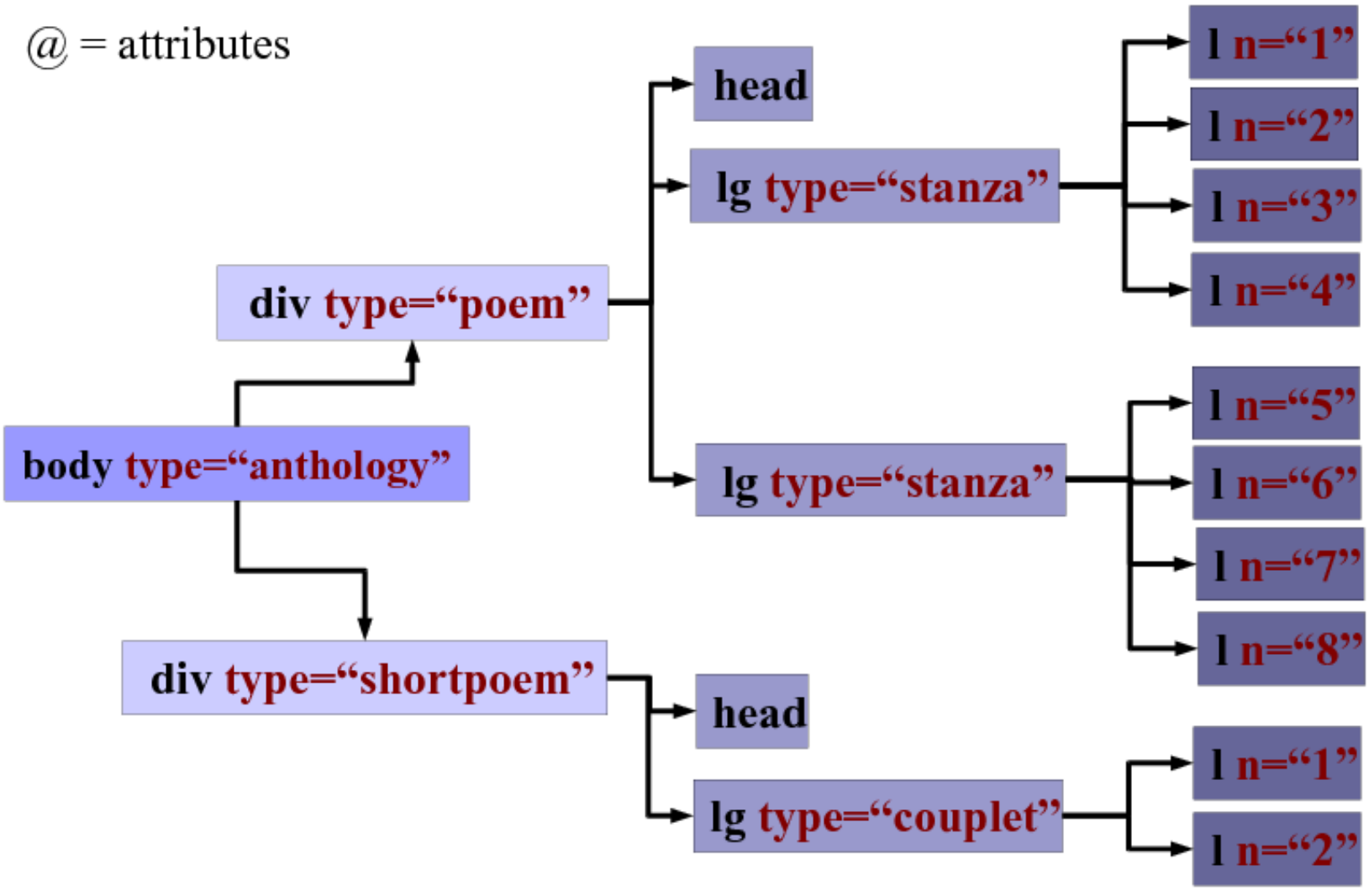
/body/div/lg


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"

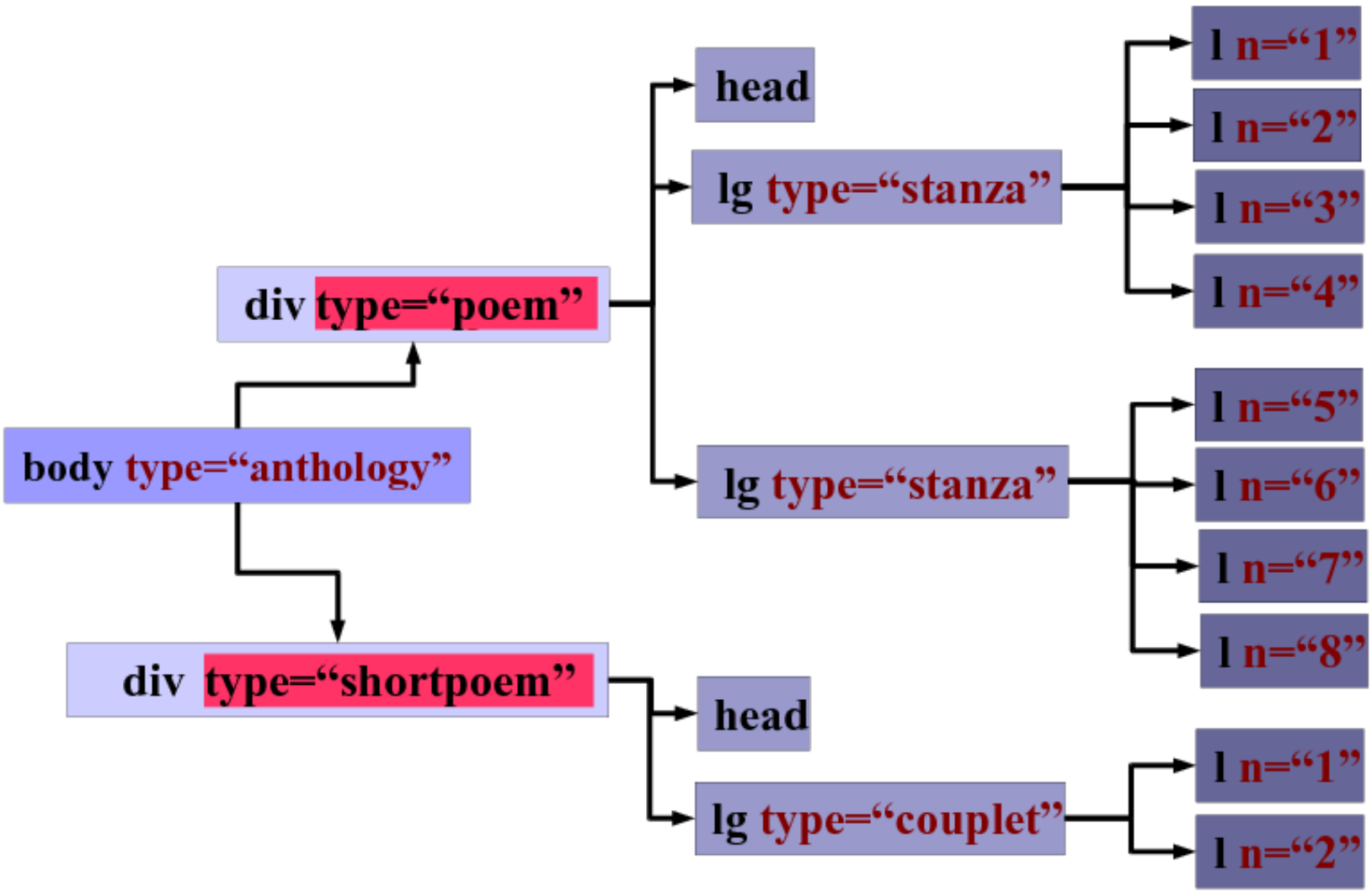
Stages on the path

- As we go along the path, we can look at other things besides XML modes ...
- we can check attributes
- and text nodes

/body/div/@type ?



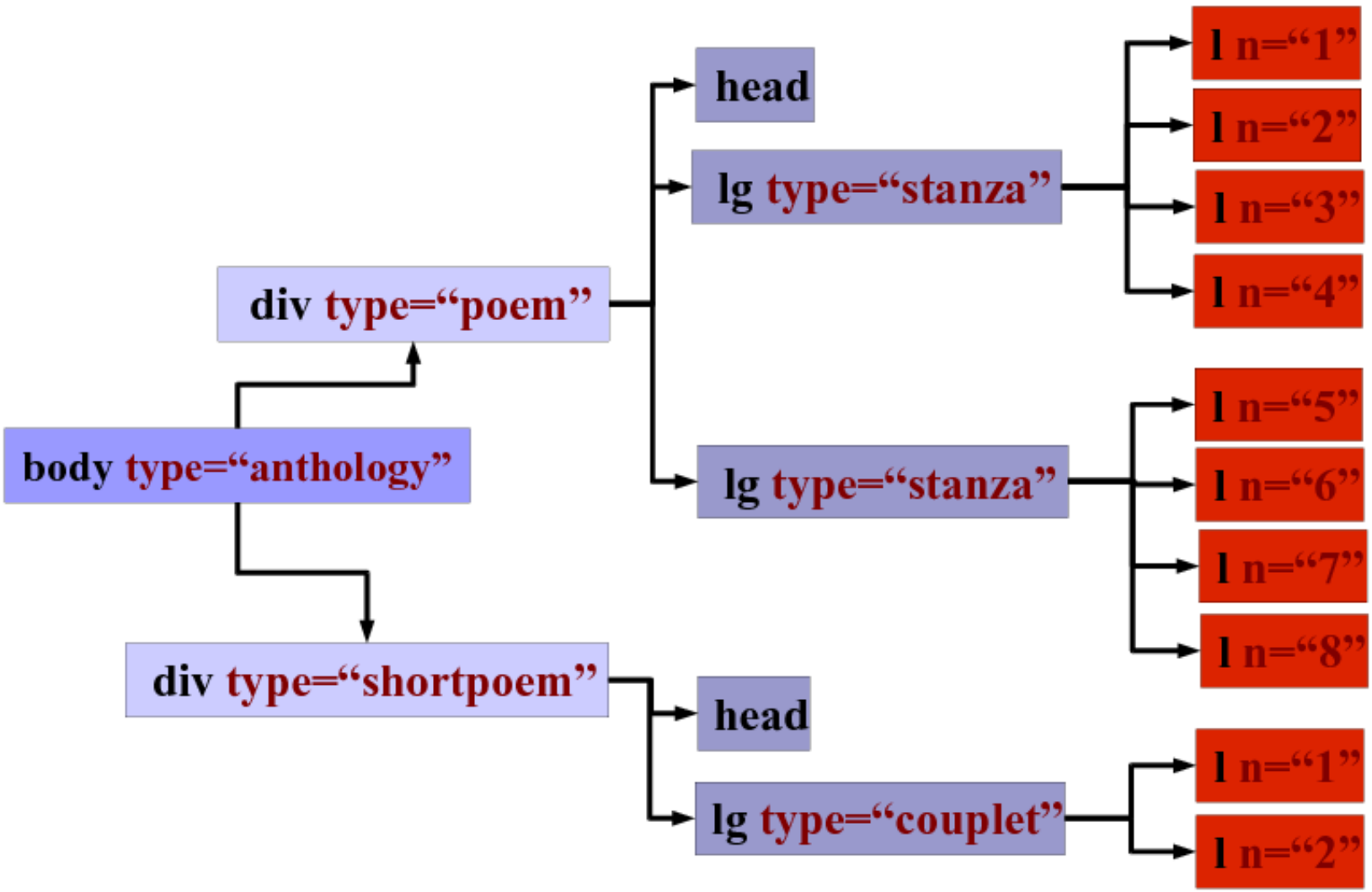
/body/div/@type



Selection

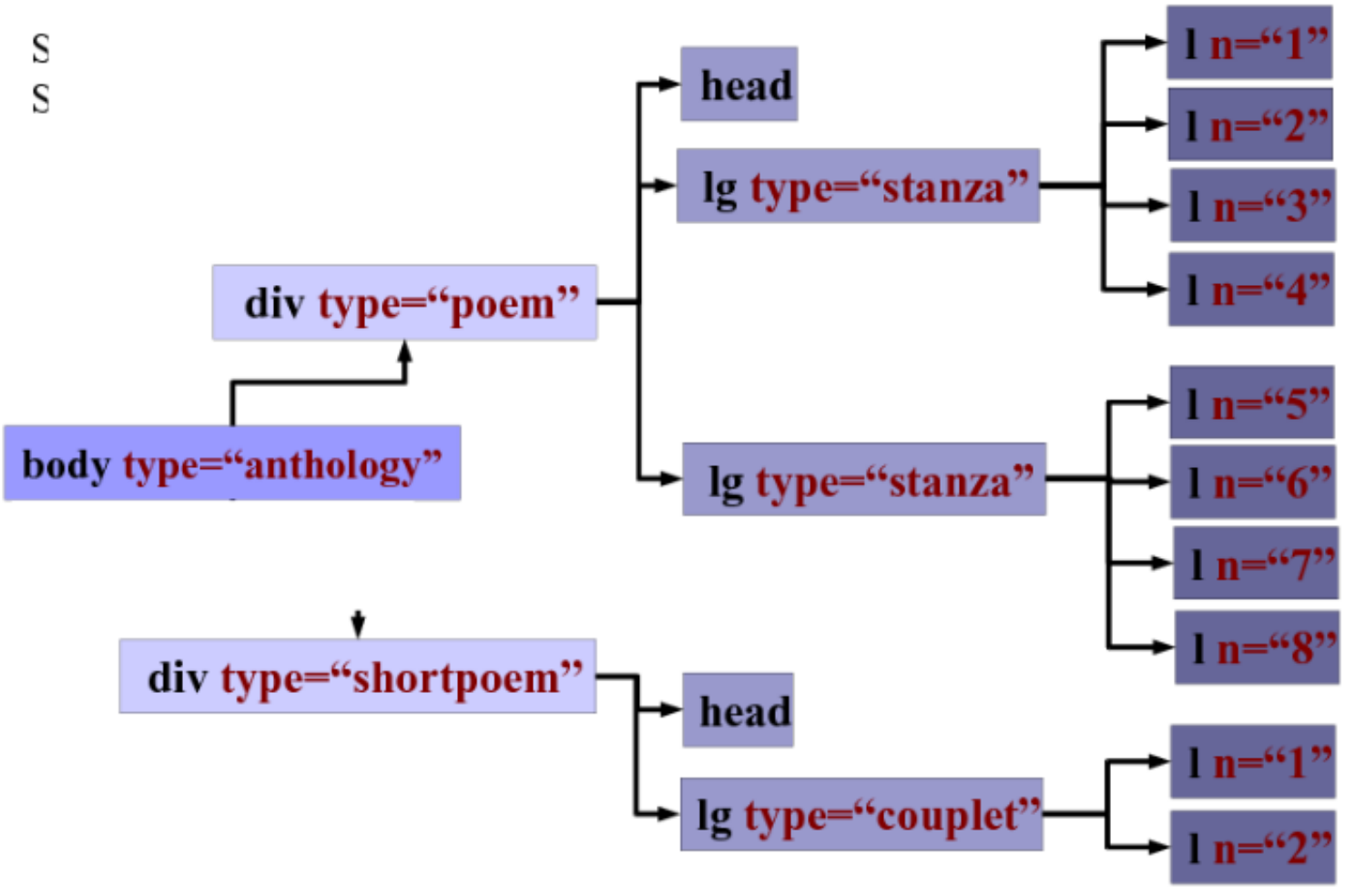
- We can select from the nodes we visit, by expressing a *restriction* using brackets [and]
- A restriction might test the value (or just the presence) of an attribute
- or the sequential position of a node in the whole tree
- or the presence of an element of a specific type at a specific place

/body/div/lg/l

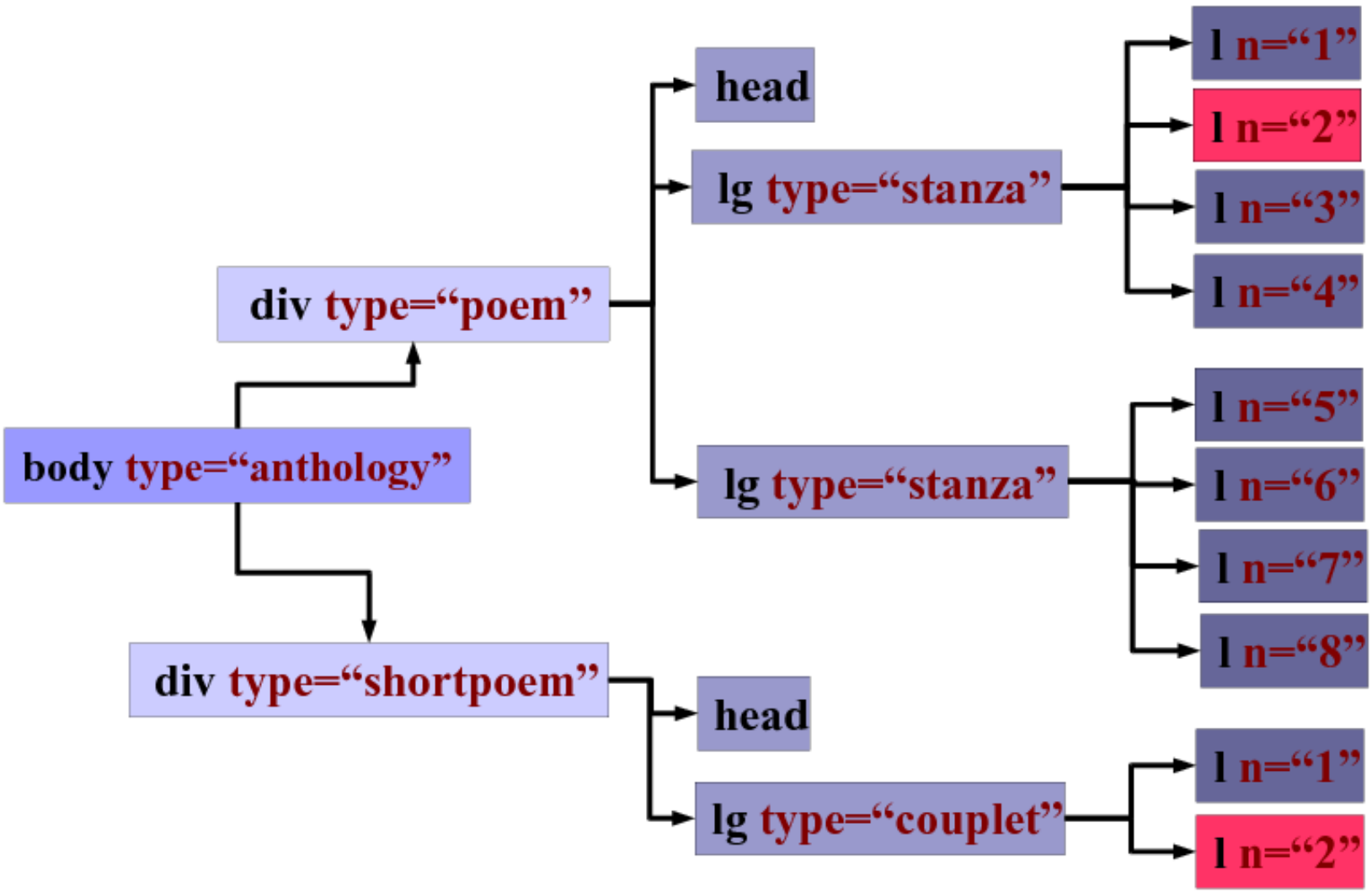

body type="anthology"
div type="poem"
head
lg type="stanza"
l n="1"
l n="2"
l n="3"
l n="4"
lg type="stanza"
l n="5"
l n="6"
l n="7"
l n="8"
div type="shortpoem"
head
lg type="couplet"
l n="1"
l n="2"

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

S
S



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



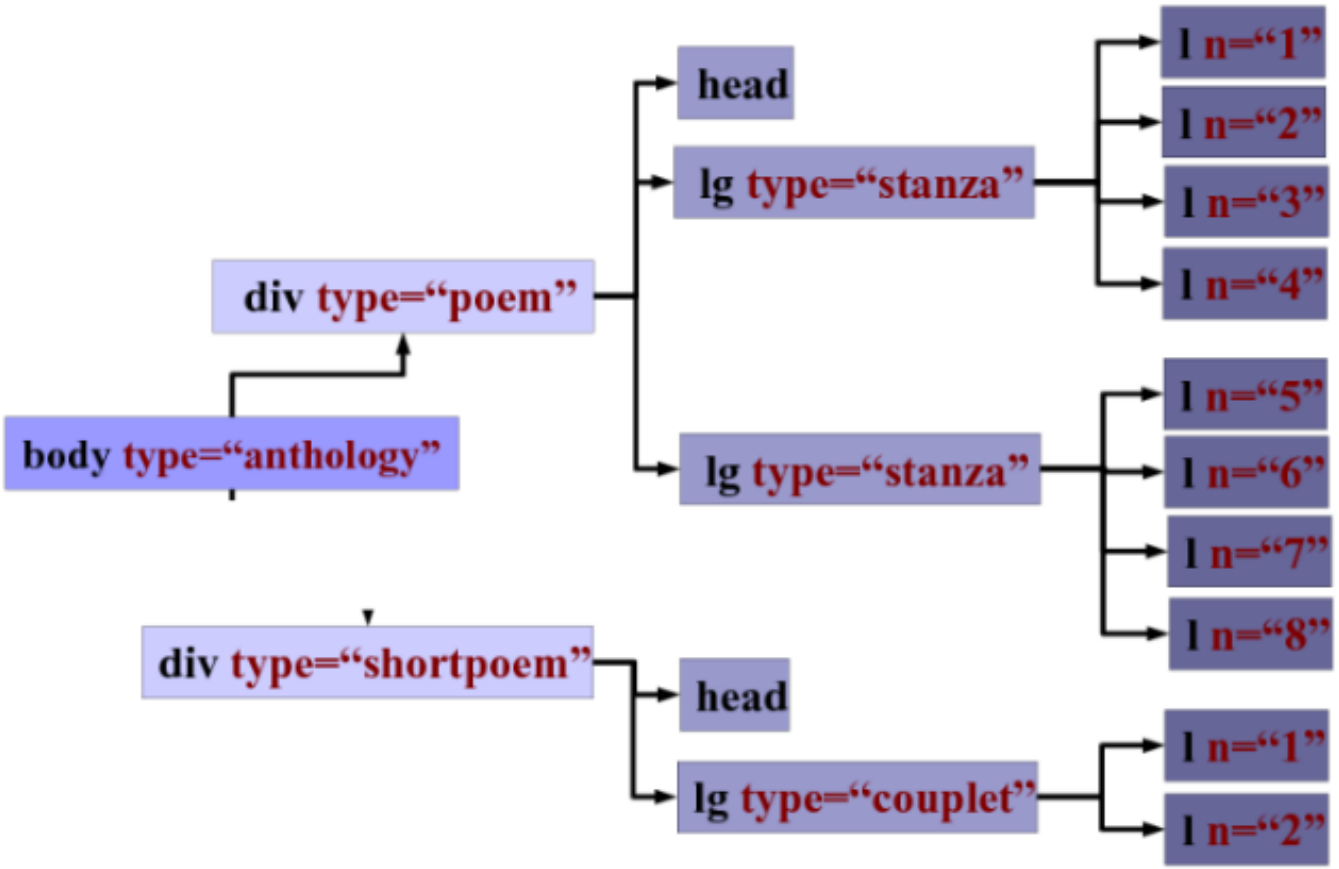
The starting point

An XPath can start from any point in the tree:

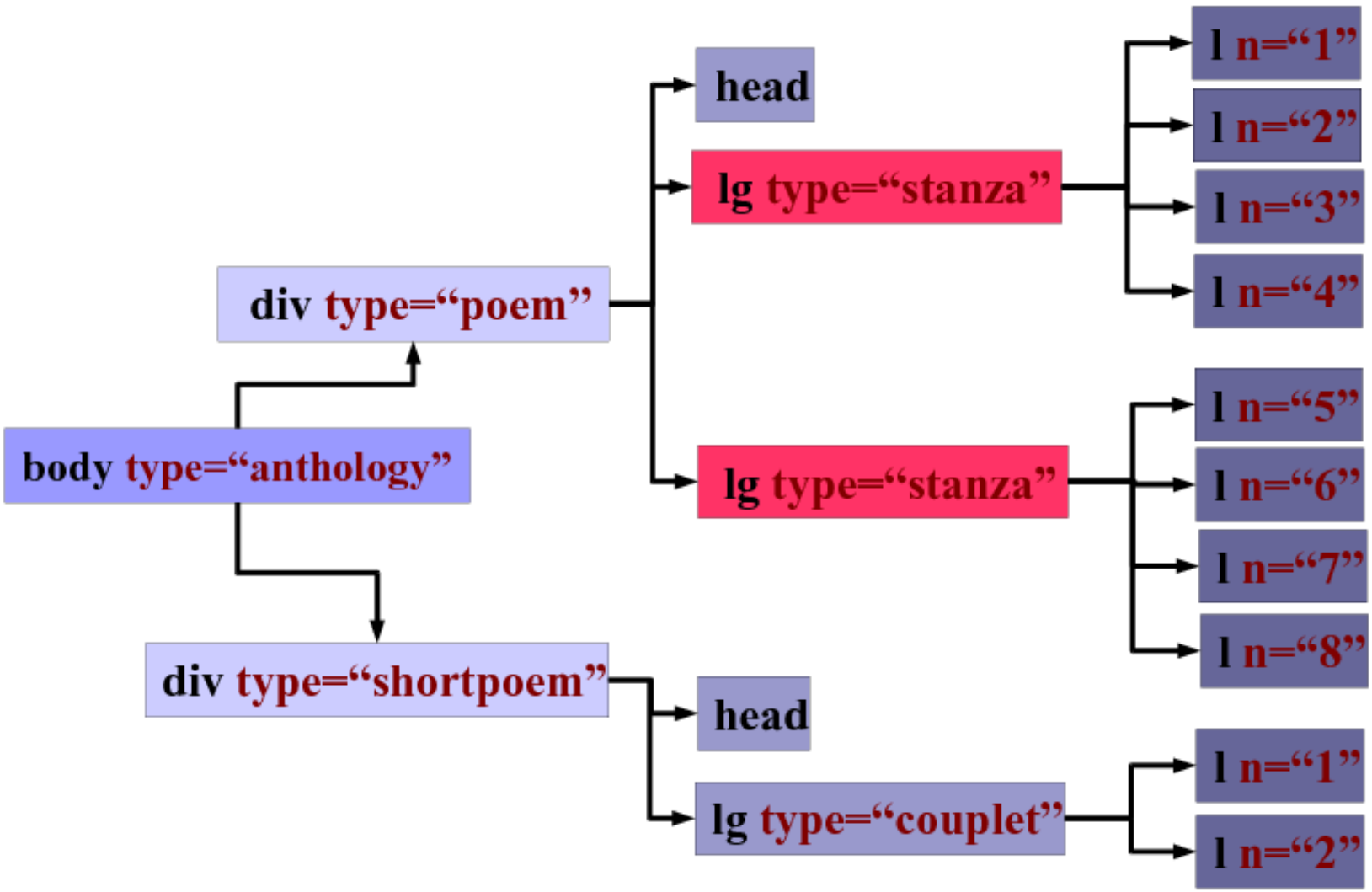
- // means ‘anywhere in the tree’
- .. means ‘my parent’

We can move freely around the hierarchy of nodes using *axes* such as ancestor::, following-sibling::, descendant:: ...

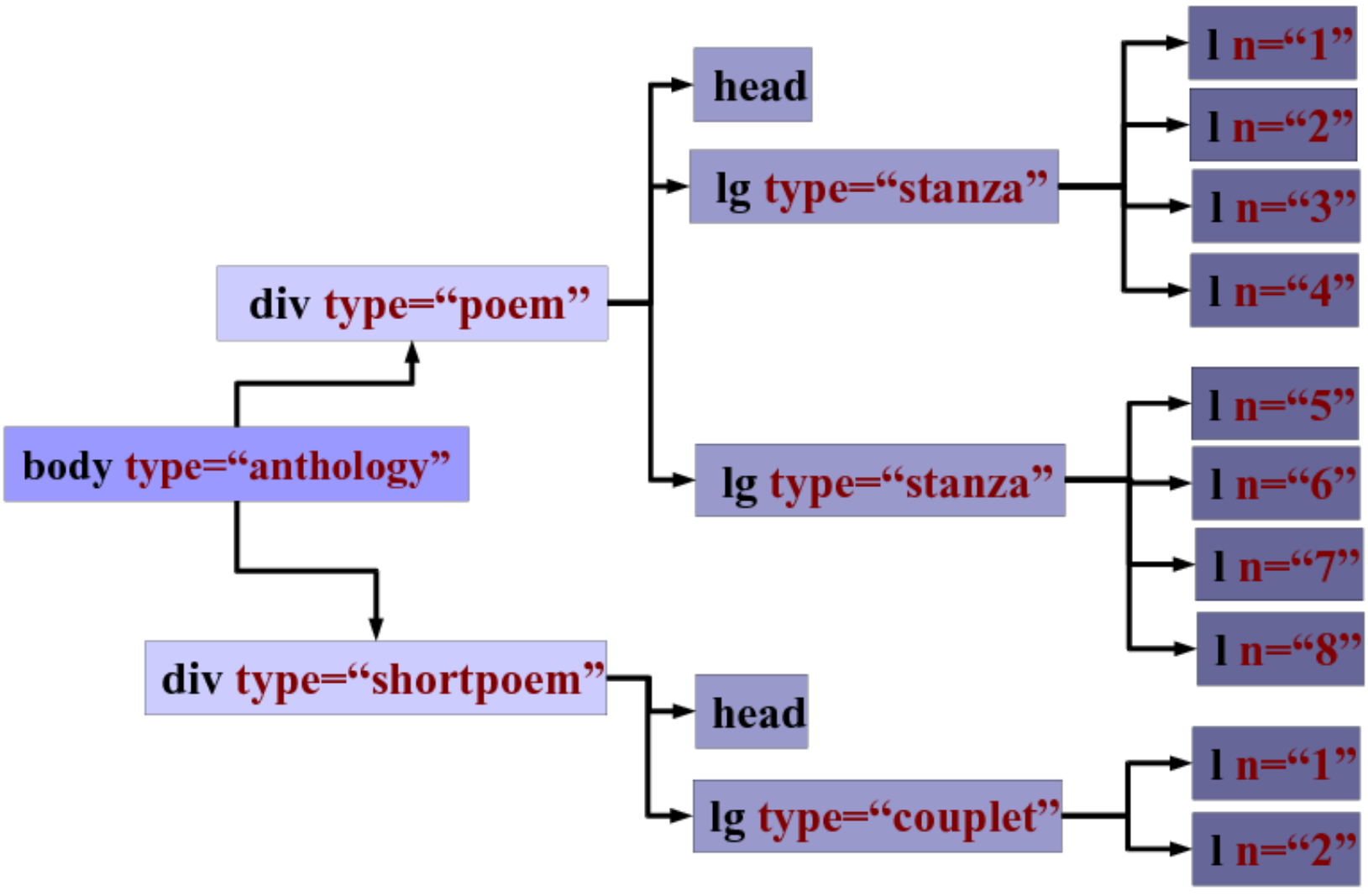
//lg[@type="stanza"] ?



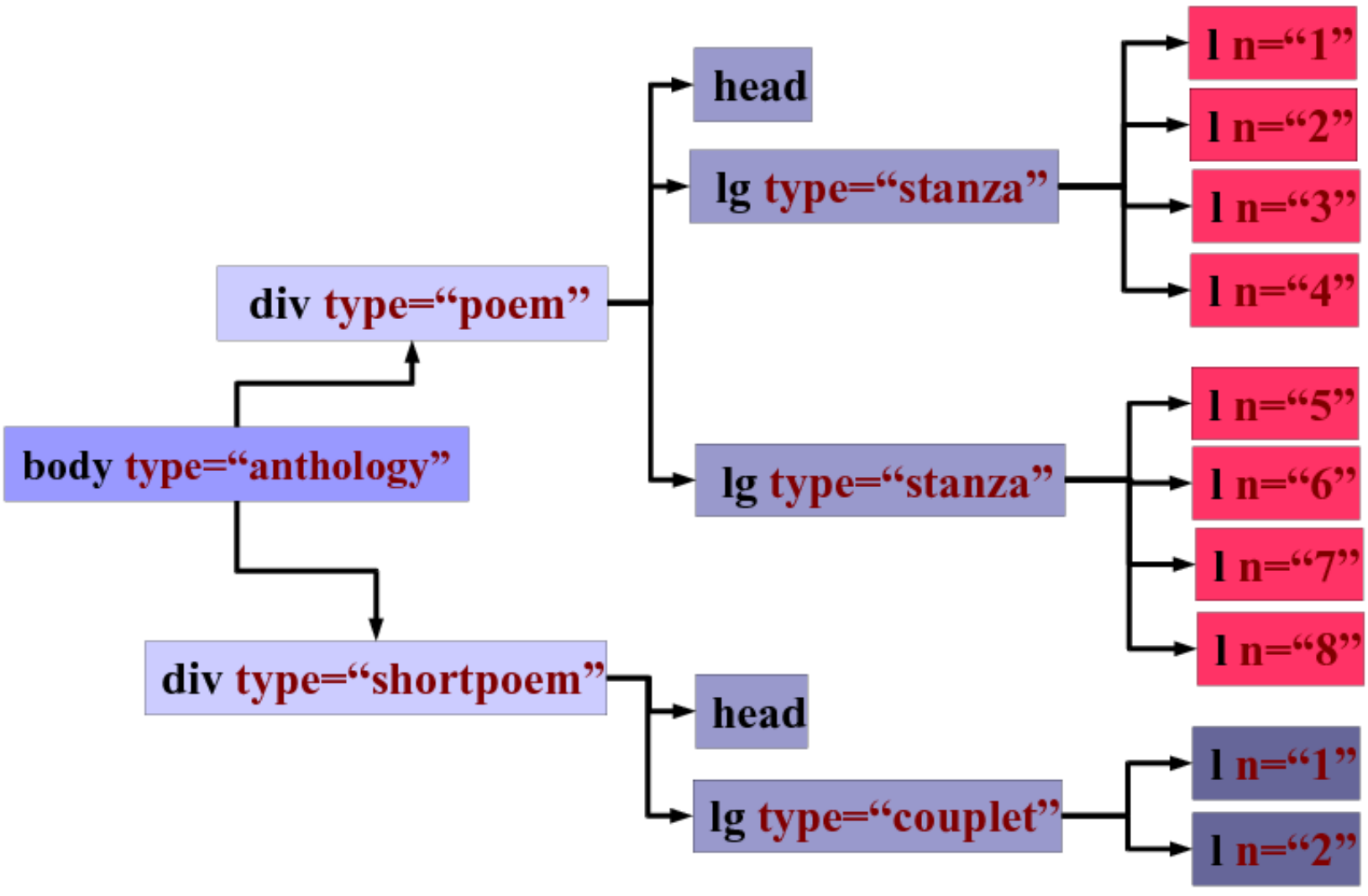
//lg[@type="stanza"]



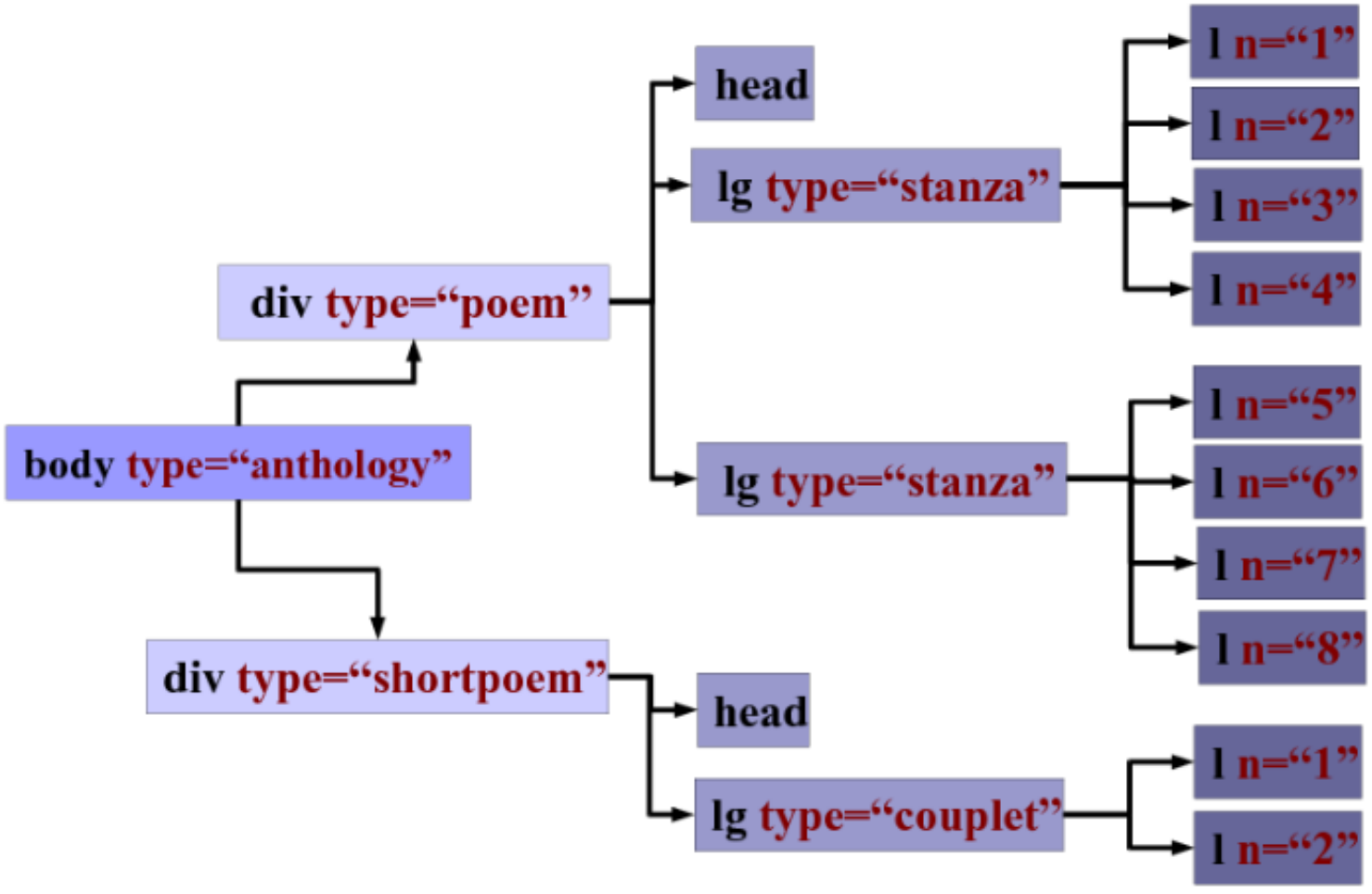
//div[@type="poem"]//l ?



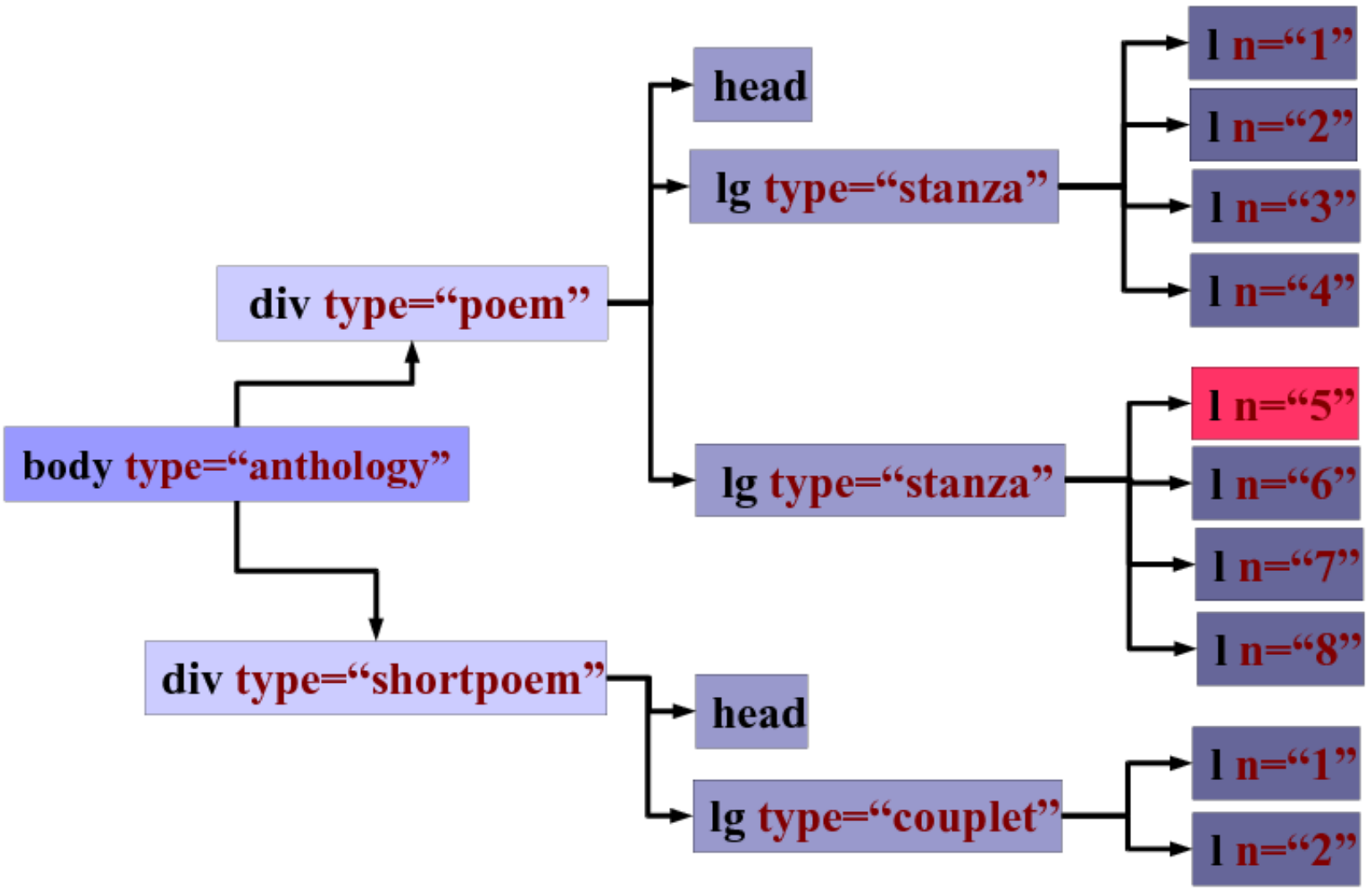
//div[@type="poem"]//l



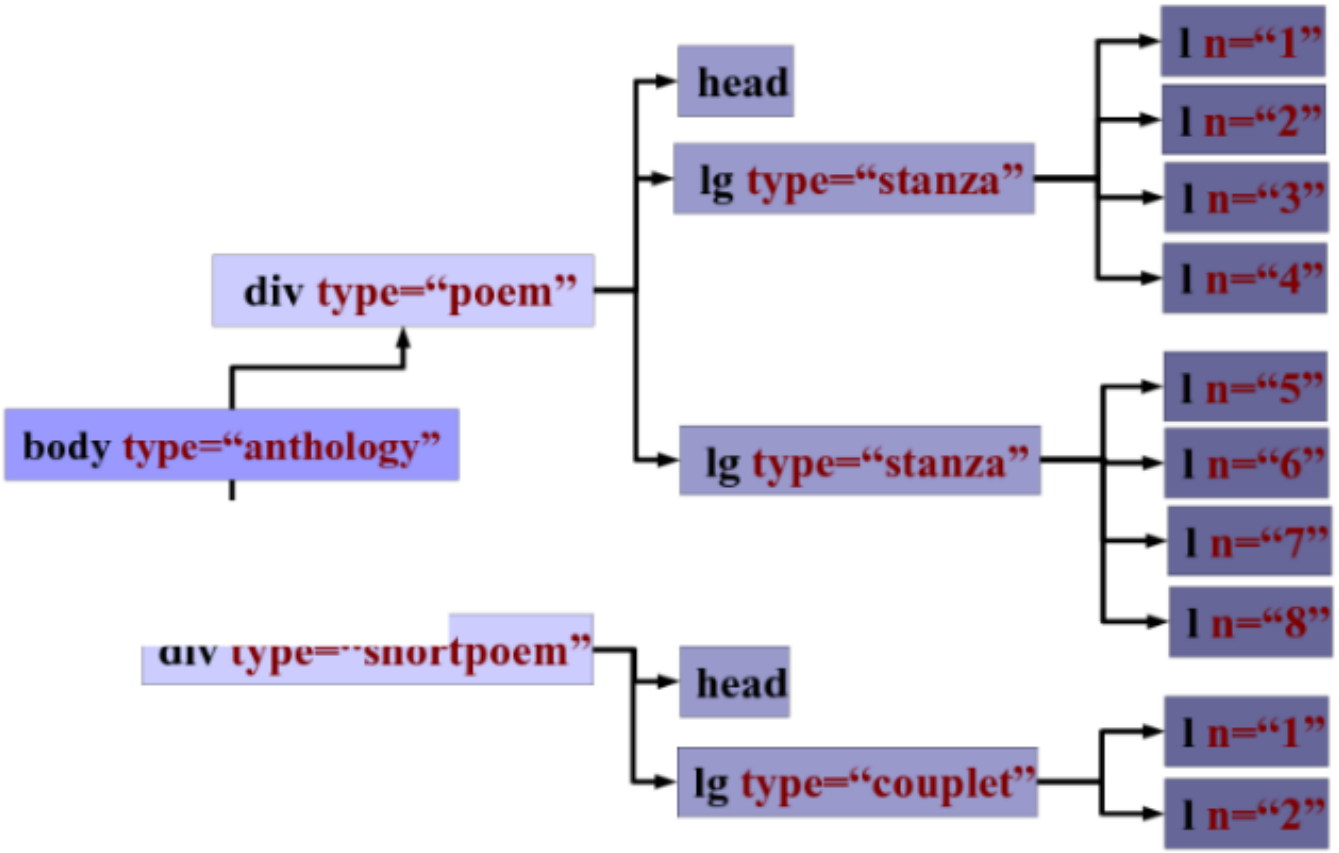
//l[5] ?


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"

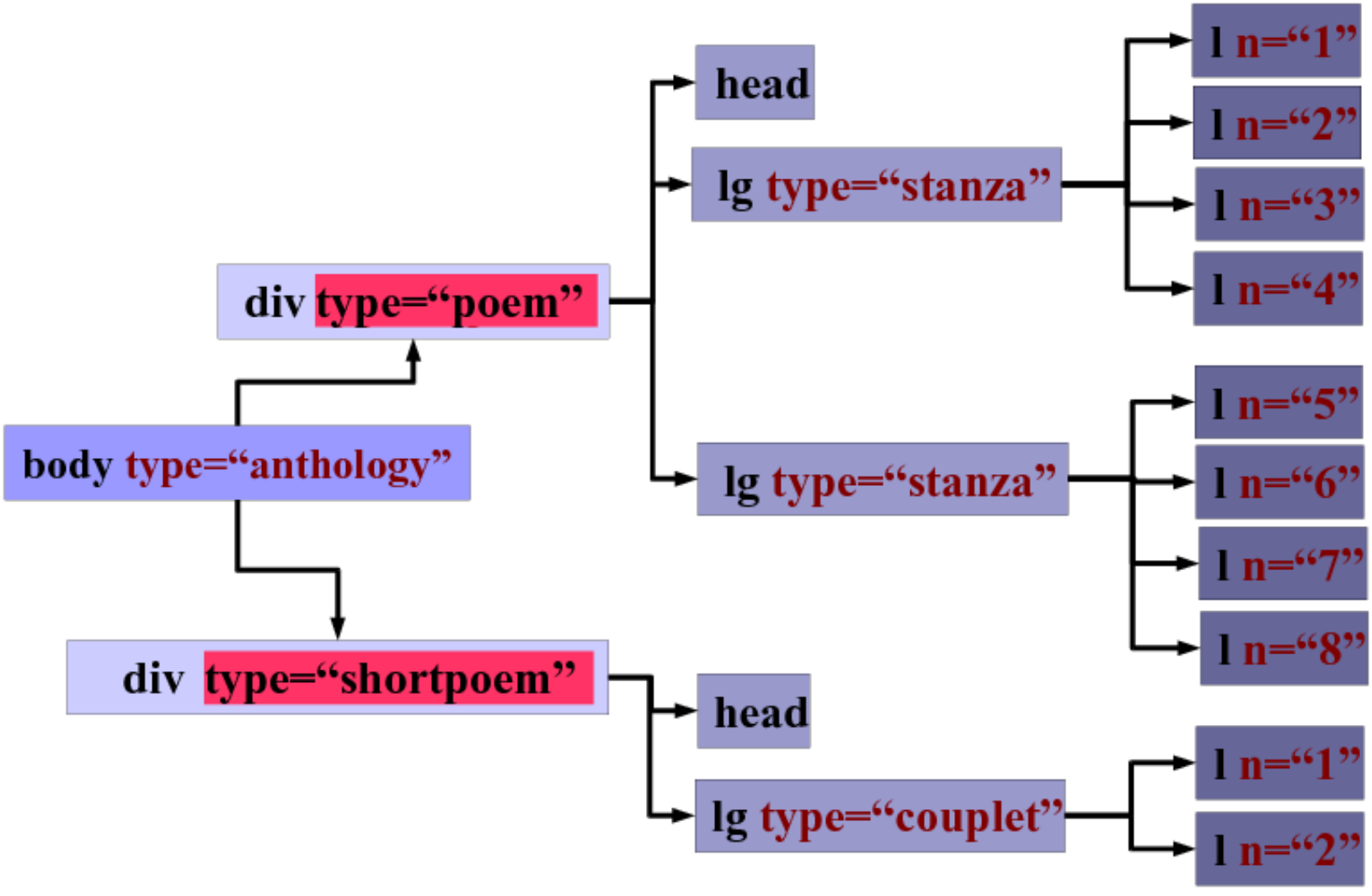
//l[5]


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"

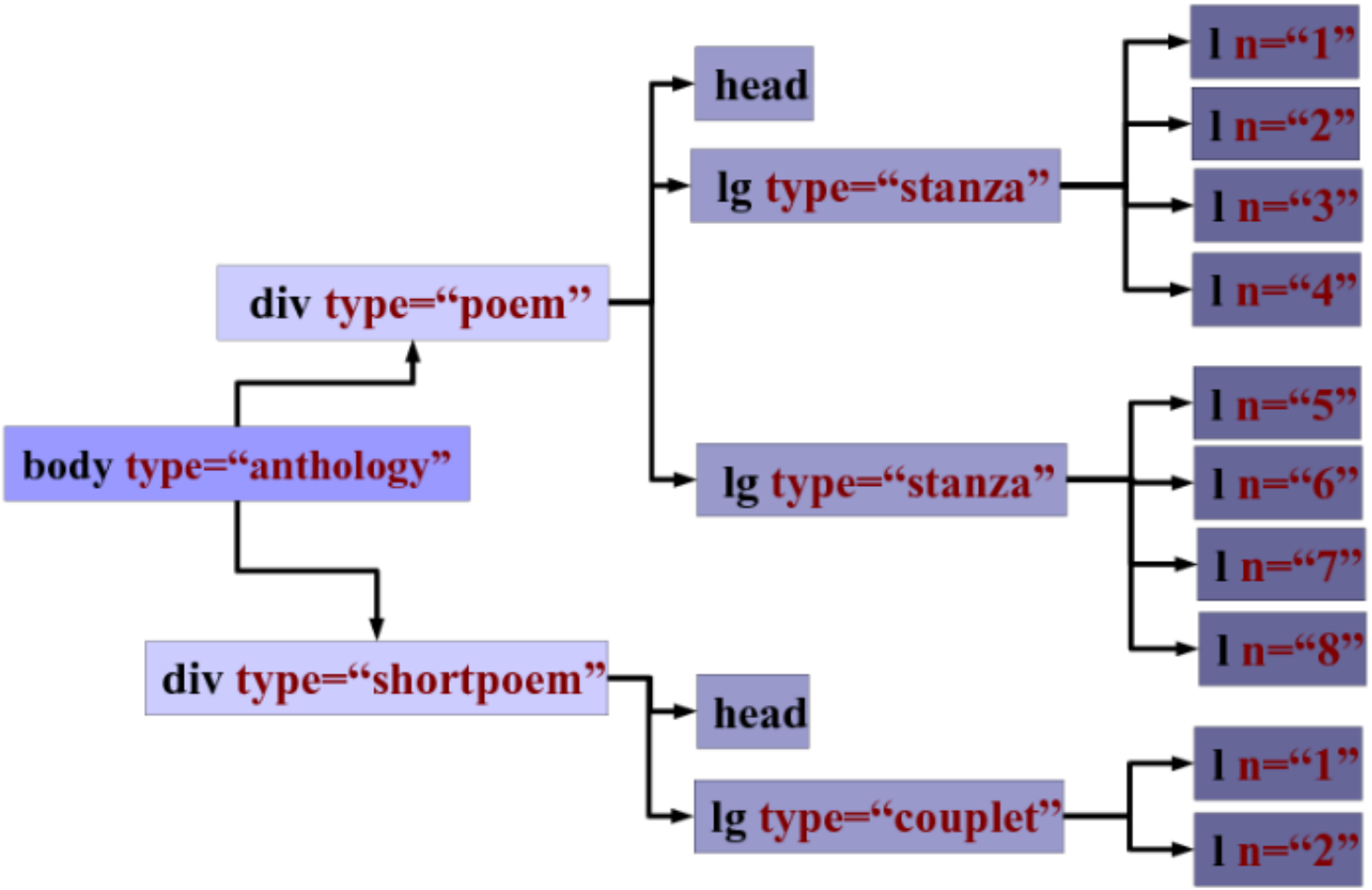
//lg/../@type ?


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"

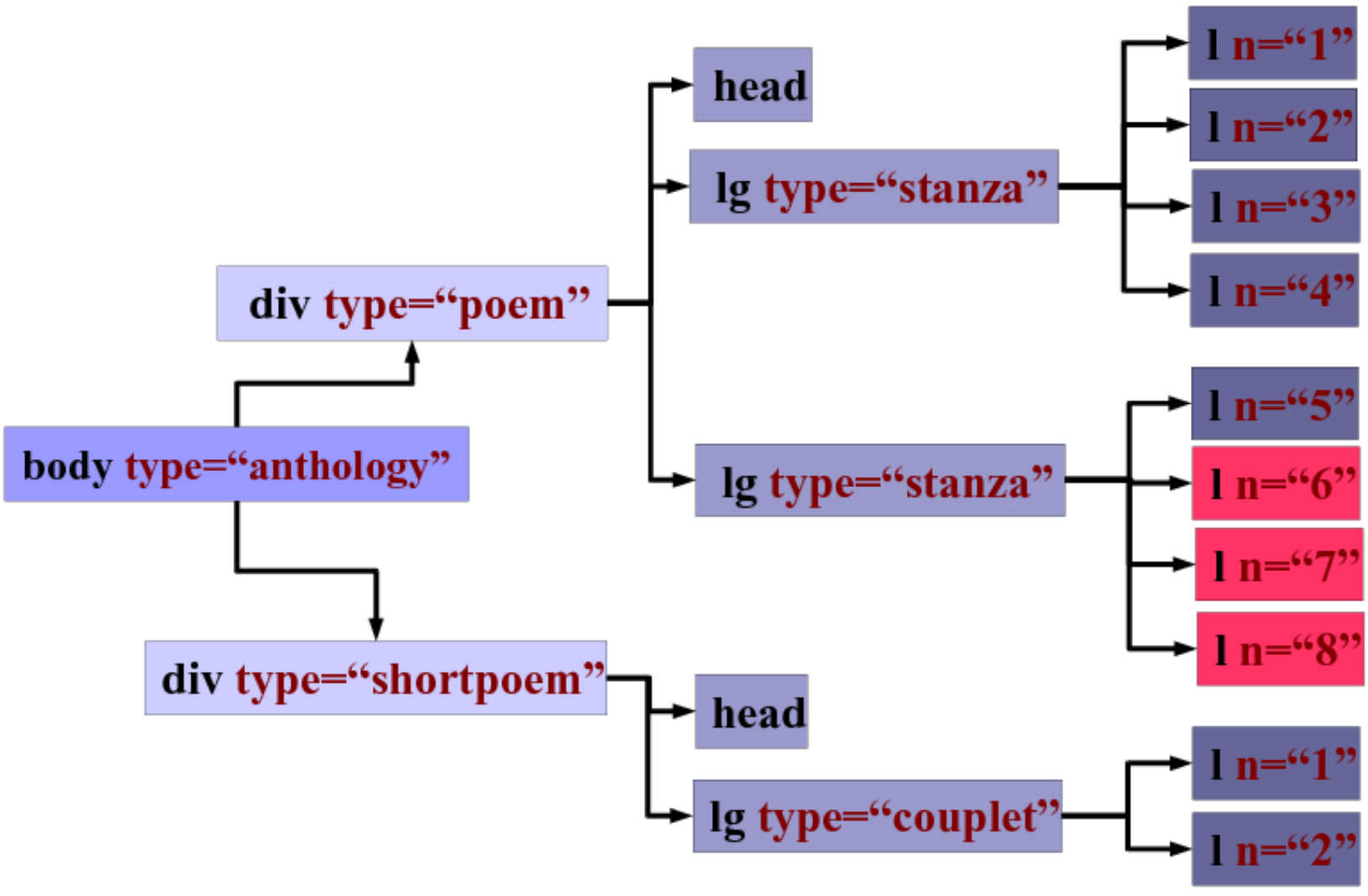
//lg/../@type


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"

XPath Functions

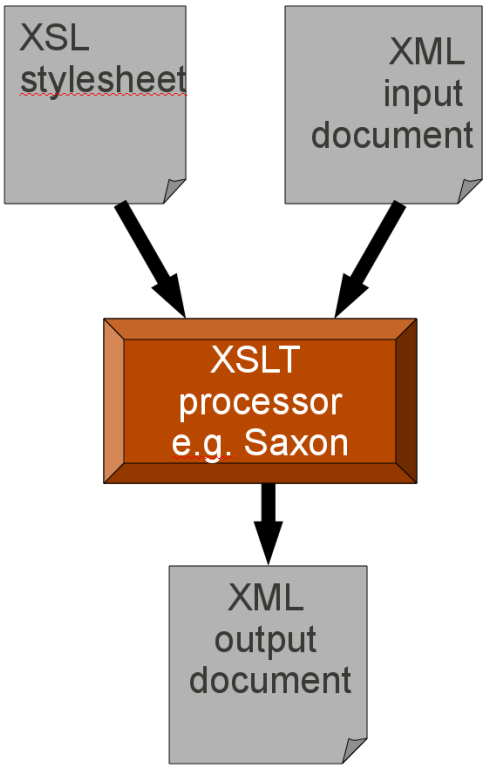
XPath also provides an extensive library of useful functions. We mention a few here :

- count(x) returns a count of the number of nodes in the tree x
- position() returns the sequential number of the current node within its context
- last() returns the sequential number of the last node within its context
- contains(x,y)returns TRUE if the string y is contained in the text node x

First Exercise

Have a look at the first part of the exercise to see whether you have understood xpath

How do you use XSLT ?



XSLT is a transformation language

And what is a ‘transformation’?

Starting from this :

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

we want to generate this :

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

So we must ....

- transform the TEI element <ref> into an (x)HTML element <a>
- transform its @*target* son attribut into an @*href* attribute

How do we express that in XSLT?

```
<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>
```

A slightly less trivial example

From this :

```
<div   type="recipe"   n="34">
  <head>Pasta for Beginners</head>
  <list>
    <item>pasta</item>
    <item>grated cheese</item>
  </list>
  <p>Boil the pasta and mix it with the cheese.</p>
</div>
```

we want to produce :

```
<html>
  <h1>34: Pasta for Beginners</h1>
  <p>Ingredients: pasta grated cheese</p>
  <p>Boil the pasta and mix it with the cheese.</p>
</html>
```

How do we express that in XSLT?

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

An XSLT stylesheet

- is an XML document, containing special elements from the XSLT namespace http://www.w3.org/1999/XSL/Transform
- The element <xsl:stylesheet> (root element for a stylesheet) can also name other namespaces, in particular a default one for elements being referenced or created; it also specifies which version of the XSLT standard is being used (1 or 2)
- The element <xsl:output> : specifies various things about the output to be generated, notable its format (HTML, XML, TEXT...), character encoding (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>
```

Ten essential XSLT elements

- <xsl:template> defines a *template*
- <xsl:apply-templates> applies templates
- <xsl:value-of> returns the value of a node
- <xsl:text> returns a bit of text
- <xsl:element>, <xsl:attribute> and <xsl:comment> create an element, attribute, or comment in the output
- <xsl:if> and <xsl:choose> conditional actions
- <xsl:for-each> looping actions
- <xsl:variable> define a variable
- <xsl:number> generate a number
- <xsl:sort> perform an ordering

<xml:template>

This element provides a template or model for the actionms which should be performed when the node or nodes specified by its *@match* attributes are found

It may contain other XSL elements, or elements from other name spaces (which will be coped to the output), or nothing at all.

```
<xsl:stylesheet   xpath-default-namespace="http://www.tei-c.org/ns/1.0"
  version="2.0">
  <xsl:template   match="div">
     <!-- .... actions for all div elements ....-->
  </xsl:template>
  <xsl:template   match="head">
     <!-- .... actions for all head elements....-->
  </xsl:template>
  <xsl:template   match="div/head">
     <!-- .... actions for head elements contained by a div....-->
  </xsl:template>
  <xsl:template   match="teiHeader"/>
</xsl:stylesheet>
```

The six golden rules of XSLT

By default, the XML tree is processed element by element, starting from the root

- If no template matches the element you are looking at, process its children
- If there are no more elements to process by rule 1, emit any text nodes contained by the element you are looking at
- An element is processed only when a template matches it
- The order of templates in a stylesheet has no significance
- Any part of the XML tree can be accessed, in any order, any number of times
- A stylesheet must contain only well formed XML

Exercise 2

However, it is much easier to understand how XSLT works by looking at a real example.

So... let's do exercise 2.

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...