XPATH

Unit-II

XPath

- A Language for Locating Nodes in XML Documents
- *XPath expressions* are written in a syntax that resembles paths in file systems
- The list of nodes located by an XPath expression is called a *Nodelist*
- XPath is used in XSL and in XQuery (a query language for XML)
- XPath includes
  - Axis navigation
  - Conditions
  - Functions

Syntax of XPath Expressions

- "/" at the beginning of an XPath expression represents the root of the document
- "/" between element names represents a parent-child relationship
- "//" represents an ancestor-descendent relationship
- "@" marks an attribute
- "[*condition*]" specifies a condition

Slashes

- A path that begins with a `/` represents an absolute path, starting from the top of the document
  - Example: `/email/message/header/from`
  - Note that even an absolute path can select *more than one* element
  - A slash by itself means "the whole document"
- A path that does *not* begin with a `/` represents a path starting from the current element
  - Example: `header/from`
- A path that begins with `//` can start from *anywhere* in the document
  - Example: `//header/from` selects every element `from` that is a child of an element `header`
  - This can be expensive, since it involves searching the entire document

Brackets and last()

- A number in brackets selects a particular matching child (counting starts from 1)
  - Example: /library/book[1] selects the first book of the library
  - Example: //chapter/section[2] selects the second section of every chapter in the XML document
  - Example: //book/chapter[1]/section[2]
  - Only *matching* elements are counted; for example, if a book has both sections and exercises, the latter are ignored when counting sections
- The function last() in brackets selects the last matching child
  - Example: /library/book/chapter[last()]
- You can even do simple arithmetic
  - Example: /library/book/chapter[last()-1]

Stars

- A star, or asterisk, is a “wild card”—it means “all the elements at this level”
  - Example: /library/book/chapter/* selects every child of every chapter of every book in the library
  - Example: //book/* selects every child of every book (chapters, tableOfContents, index, etc.)
  - Example: /*/*/*/paragraph selects every paragraph that has exactly three ancestors
  - Example: //* selects every element in the entire document

Attributes I

- You can select attributes by themselves, or elements that have certain attributes
  - Remember: an attribute consists of a name-value pair, for example in <chapter num="5">, the attribute is named num
  - To choose the attribute itself, prefix the name with @
  - Example: @num will choose every *attribute* named num
  - Example: //@* will choose *every attribute, everywhere* in the document
- To choose *elements* that have a given attribute, put the attribute name in square brackets
  - Example: //chapter[@num] will select every chapter element (anywhere in the document) that has an attribute named num

Attributes II

- //chapter[@num] selects every chapter element with an attribute num
- //chapter[not(@num)] selects every chapter element that does *not* have a num attribute
- //chapter[@*] selects every chapter element that has *any* attribute
- //chapter[not(@*)] selects every chapter element with *no* attributes

Values of attributes

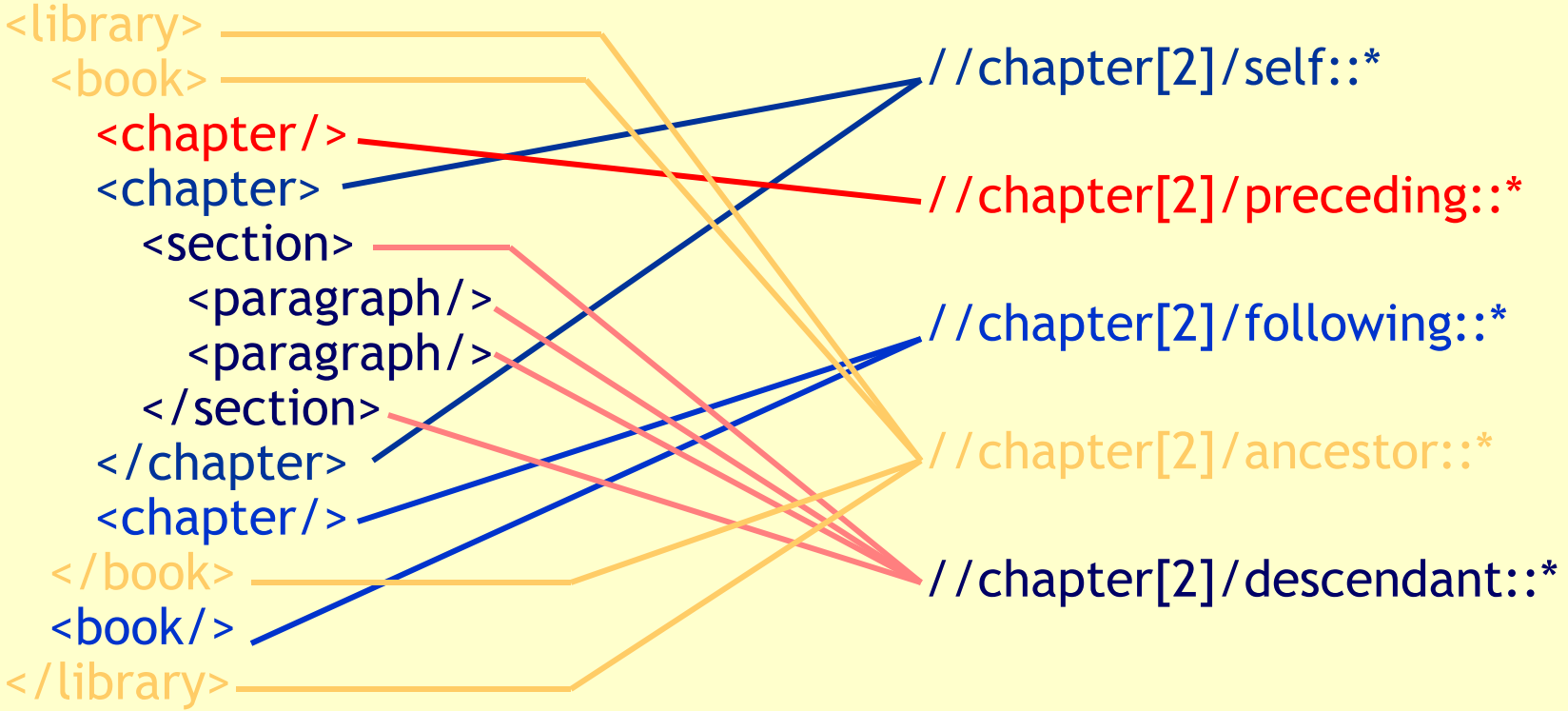
- //chapter[@num='3'] selects every chapter element with an attribute num with value 3
- //chapter[not(@num)] selects every chapter element that does *not* have a num attribute
- //chapter[@*] selects every chapter element that has *any* attribute
- //chapter[not(@*)] selects every chapter element with *no* attributes
- The normalize-space() function can be used to remove leading and trailing spaces from a value before comparison
  - Example: //chapter[normalize-space(@num)="3"]

Axes

- An axis (plural axes) is a set of nodes relative to a given node; X::Y means "choose Y from the X axis"
  - self:: is the set of current nodes (not too useful)
    - self::node() is the current node
  - child:: is the default, so /child::X is the same as /X
  - parent:: is the parent of the current node
  - ancestor:: is all ancestors of the current node, up to and including the root
  - descendant:: is all descendants of the current node (Note: never contains attribute or namespace nodes)
  - preceding:: is everything before the current node in the entire XML document
  - following:: is everything after the current node in the entire XML document

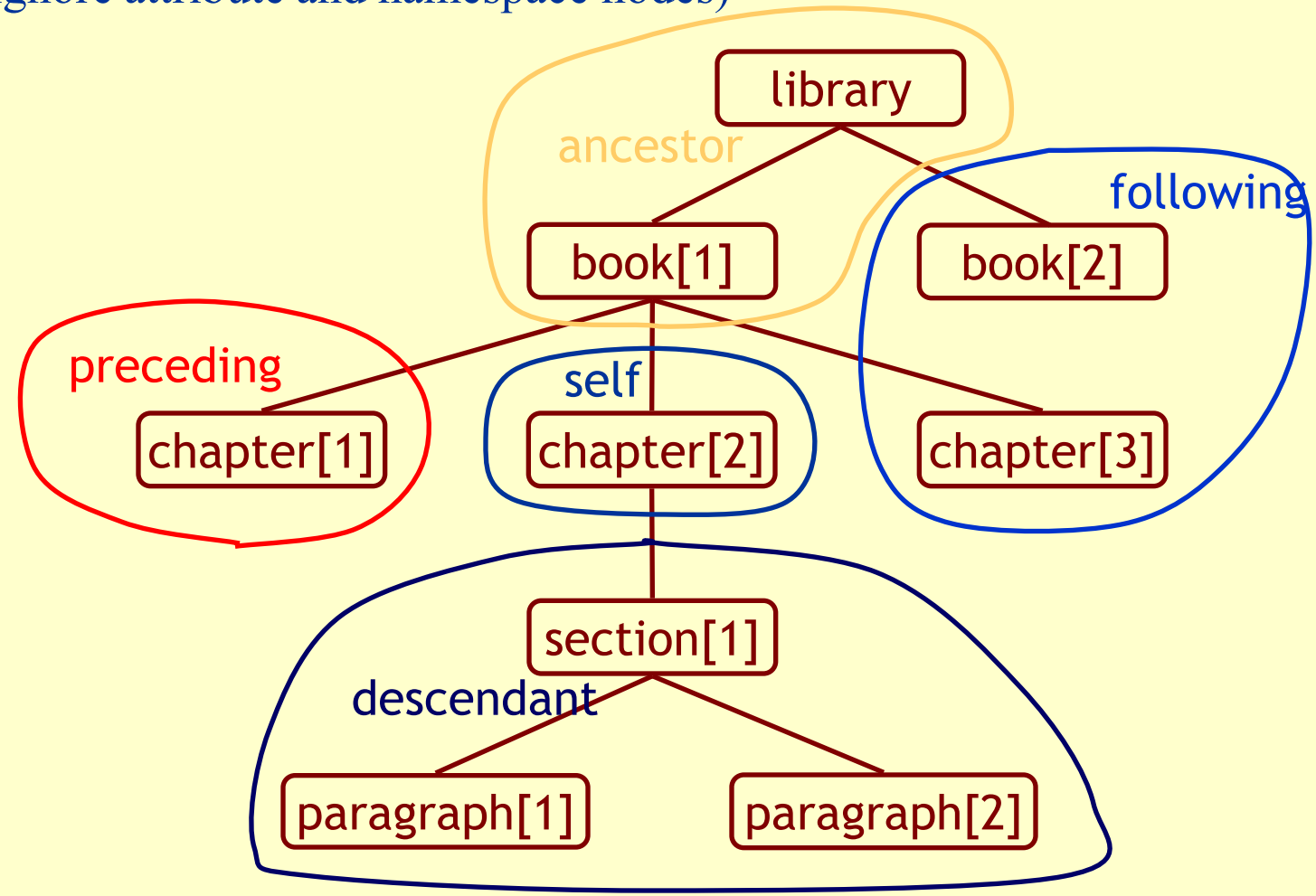
Axes (outline view)

Starting from a given node, the self, preceding, following, ancestor, and descendant axes form a partition of all the nodes (if we ignore attribute and namespace nodes)



Axes (tree view)

- Starting from a given node, the self, ancestor, descendant , preceding, and following axes form a *partition* of all the nodes (if we ignore attribute and namespace nodes)



Axis examples

- //book/descendant::* is all descendants of every book
- //book/descendant::section is all section descendants of every book
- //parent::* is every element that is a parent, i.e., is not a leaf
- //section/parent::* is every parent of a section element
- //parent::chapter is every chapter that is a parent, i.e., has children
- /library/book[3]/following::* is everything after the third book in the library

More axes

- ancestor-or-self:: ancestors plus the current node
- descendant-or-self:: descendants plus the current node
- attribute:: is all attributes of the current node
- namespace:: is all namespace nodes of the current node
- preceding:: is everything before the current node in the entire XML document
- following-sibling:: is all siblings after the current node

- Note: preceding-sibling:: and following-sibling:: do not apply to attribute nodes or namespace nodes

Abbreviations for axes

(none)	is the same as	child::
@	is the same as	attribute::
.	is the same as	self::node()
.//X	is the same as	self::node()/descendant-or-self::node()/child::X
..	is the same as	parent::node()
../X	is the same as	parent::node()/child::X
//	is the same as	/descendant-or-self::node()/
//X	is the same as	/descendant-or-self::node()/child::X

Arithmetic expressions

+	add
-	subtract
*	multiply
div	(not /) divide
mod	modulo (remainder)

Equality tests

- = means “equal to” (Notice it’s *not* ==)
- != means “not equal to”
- But it’s not that simple!
  - *value* = *node-set* will be true if the *node-set contains any node* with a value that matches *value*
  - *value* != *node-set* will be true if the *node-set contains any node* with a value that does *not* match *value*

Other boolean operators

- and (infix operator)
- or (infix operator)
  - Example: count = 0 or count = 1
- not() (function)

- The following are used for *numerical* comparisons only:
- < "less than" Some places may require <
- <= "less than or equal to" Some places may require <=
- > "greater than" Some places may require >
- >= "greater than or equal to" Some places may require >=

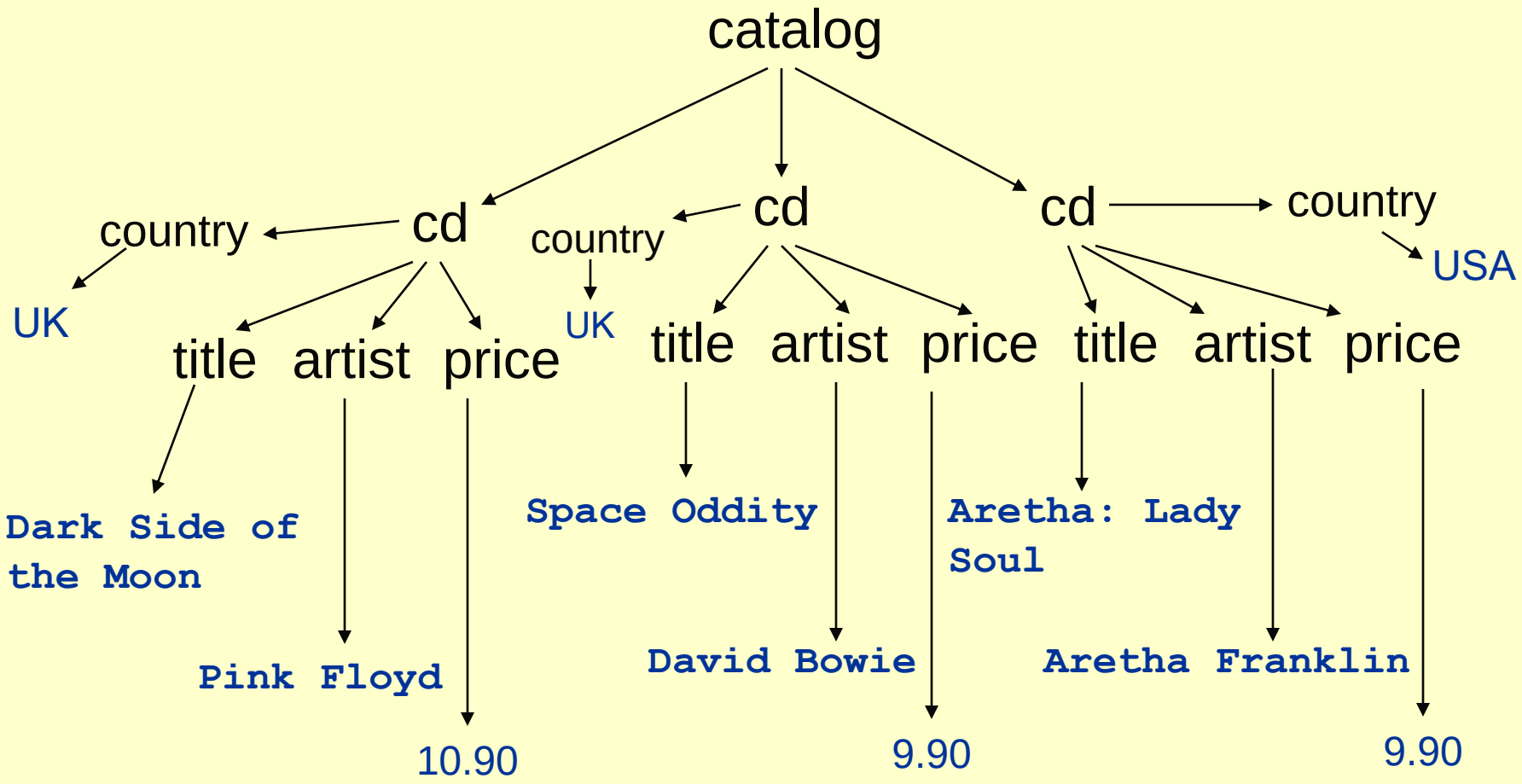
Some XPath functions

- XPath contains a number of functions on node sets, numbers, and strings; here are a few of them:
  - count(*elem*) counts the number of selected elements
    - Example: //chapter[count(section)=1] selects chapters with exactly two section children
  - name() returns the name of the element
    - Example: //*[name()='section'] is the same as //section
  - starts-with(*arg1*, *arg2*) tests if *arg1* starts with *arg2*
    - Example: //*[starts-with(name(), 'sec']
  - contains(*arg1*, *arg2*) tests if *arg1* contains *arg2*
    - Example: //*[contains(name(), 'ect']

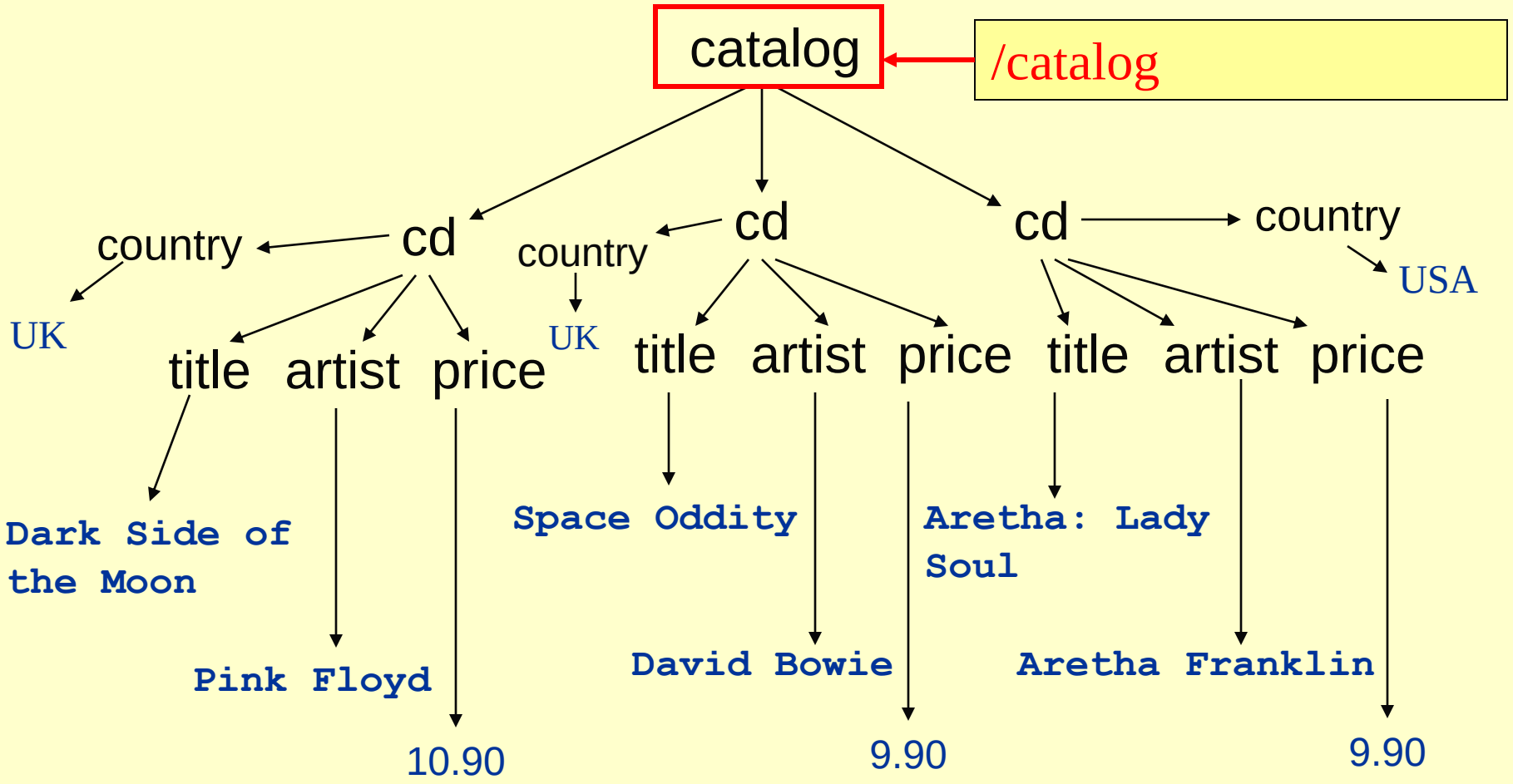
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<catalog>
    <cd country="UK">
      <title>Dark Side of the Moon</title>
      <artist>Pink Floyd</artist>
      <price>10.90</price>
    </cd>
    <cd country="UK">
        <title>Space Oddity</title>
        <artist>David Bowie</artist>
        <price>9.90</price>
    </cd>
    <cd country="USA">
        <title>Aretha: Lady Soul</title>
        <artist>Aretha Franklin</artist>
        <price>9.90</price>
   </cd>
</catalog>
```

An XML document

catalog.xml

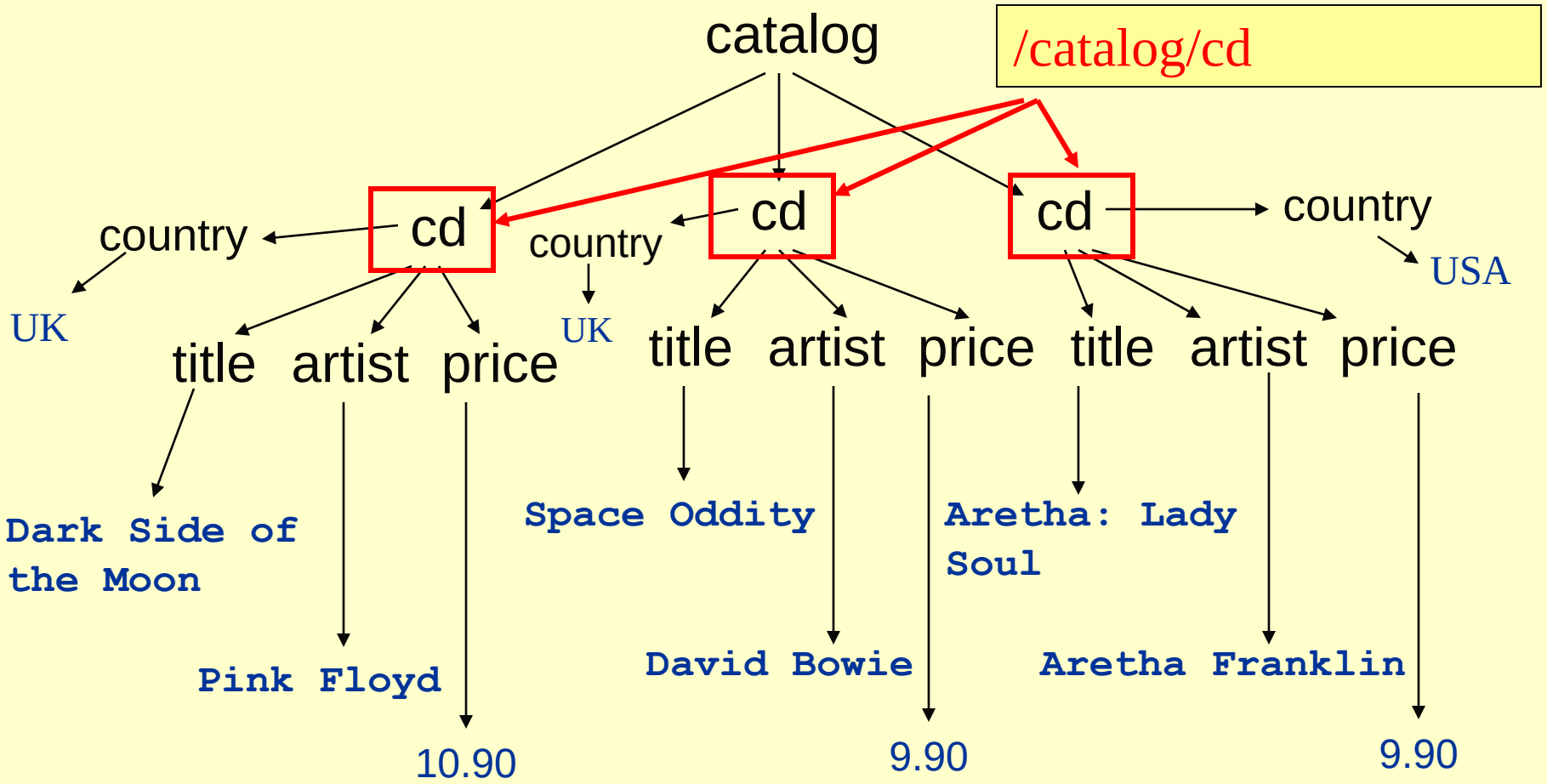

catalog
country
cd
country
cd
cd
country
UK
UK
USA
title
artist
price
title
artist
price
title
artist
price
Dark Side of the Moon
Pink Floyd
10.90
Space Oddity
David Bowie
9.90
Aretha: Lady Soul
Aretha Franklin
9.90

catalog.xml



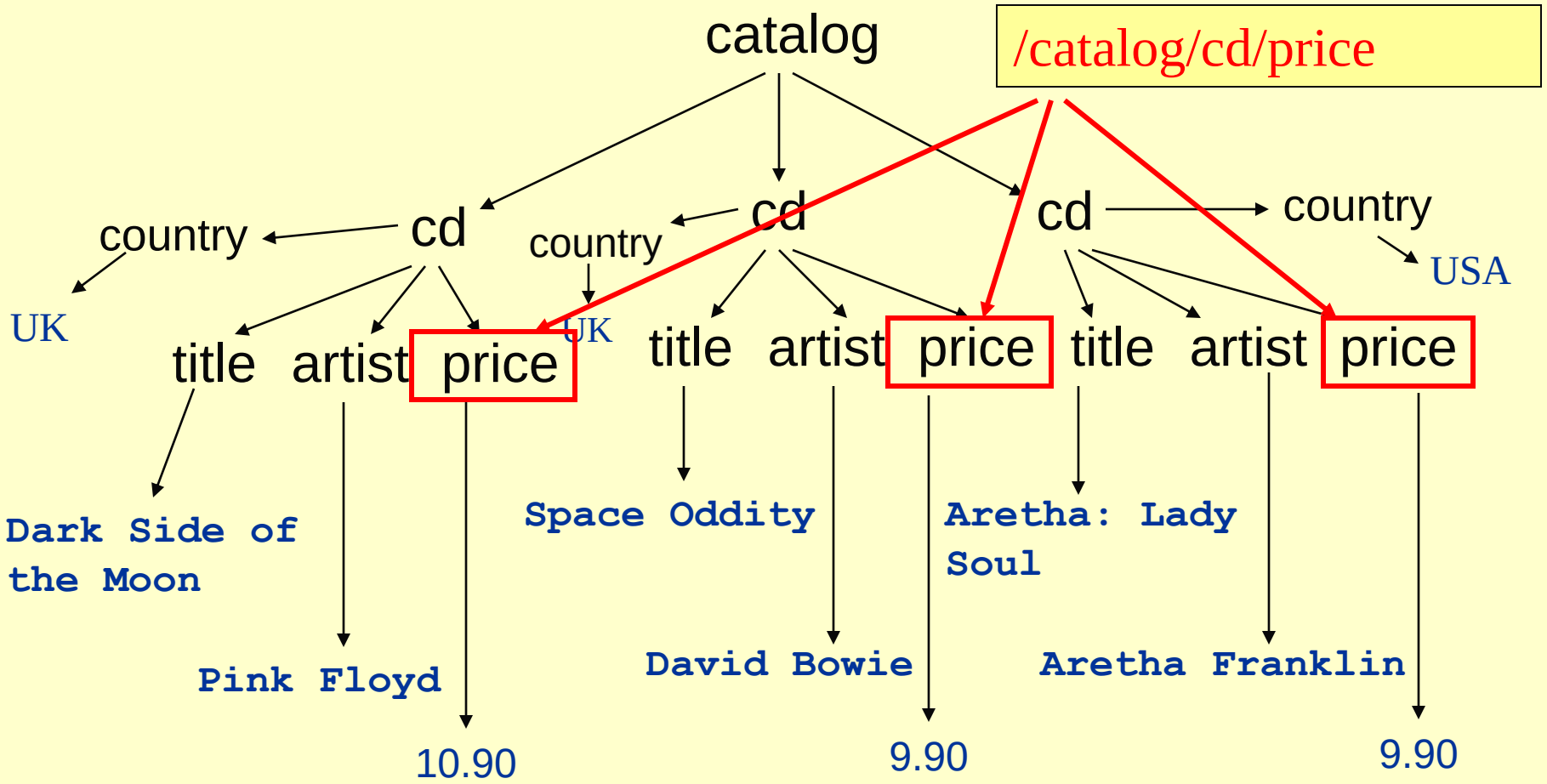
Getting the root element of the document

catalog.xml



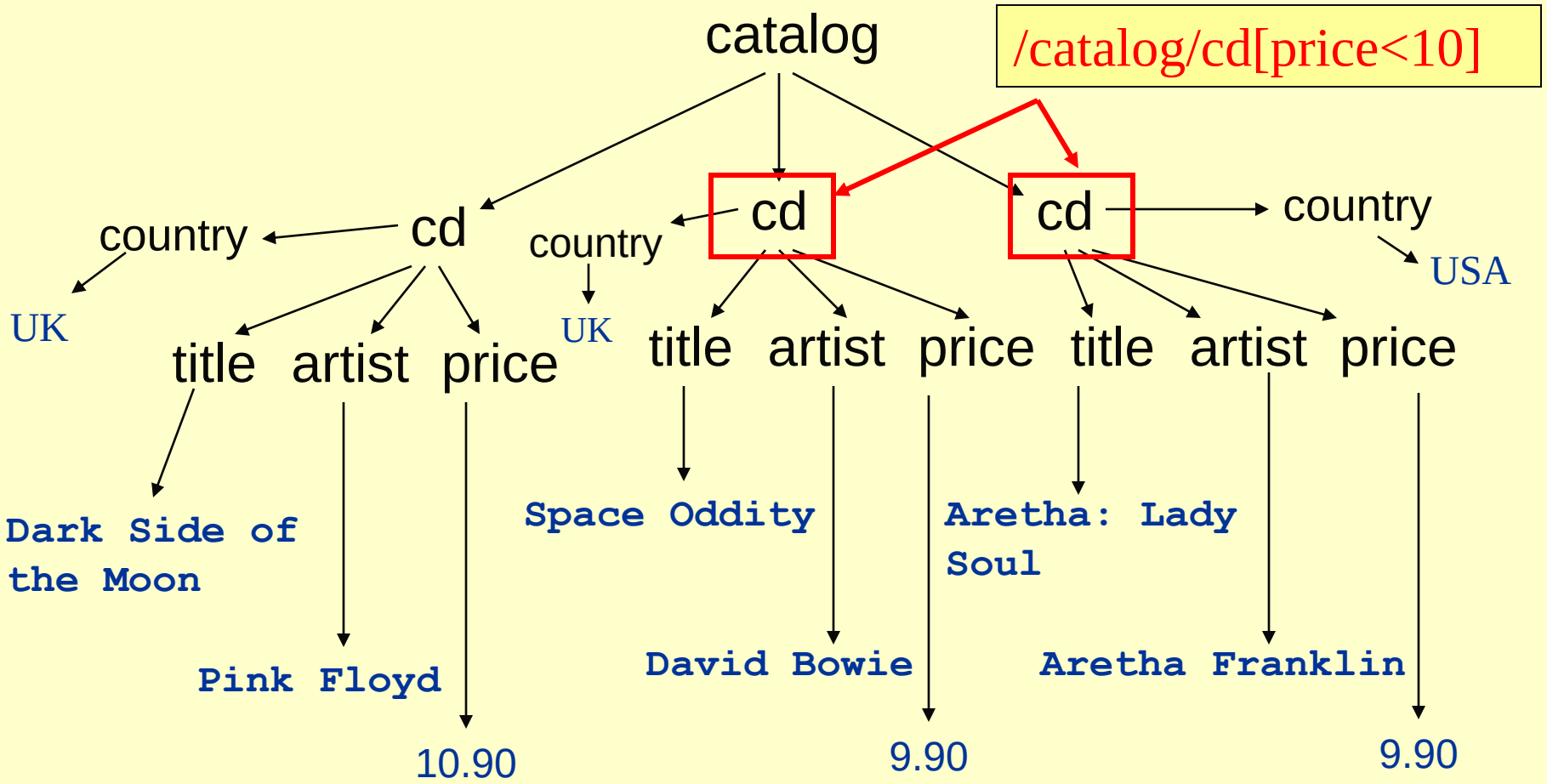
Finding child nodes

catalog.xml

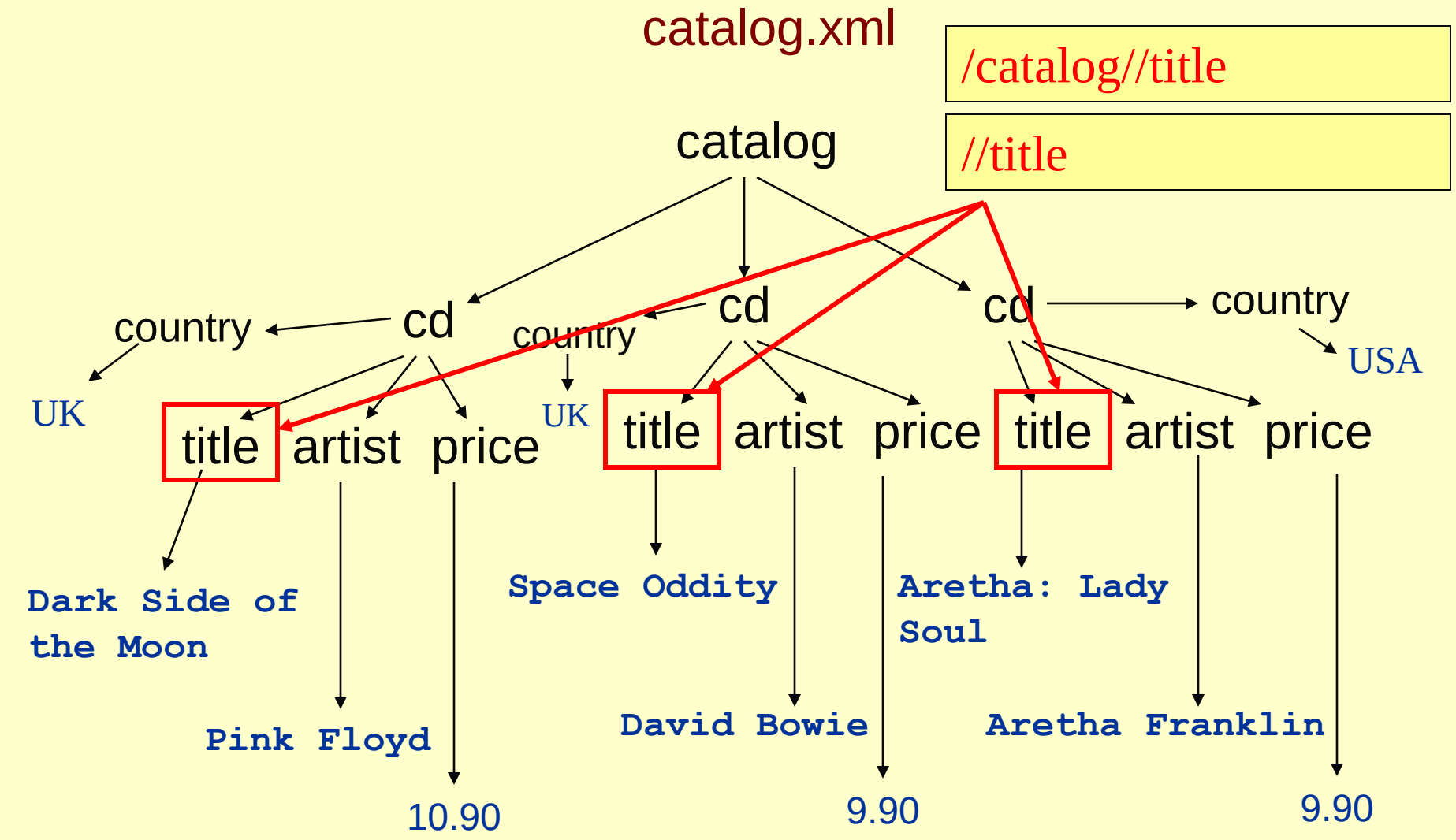


Finding descendent nodes

catalog.xml

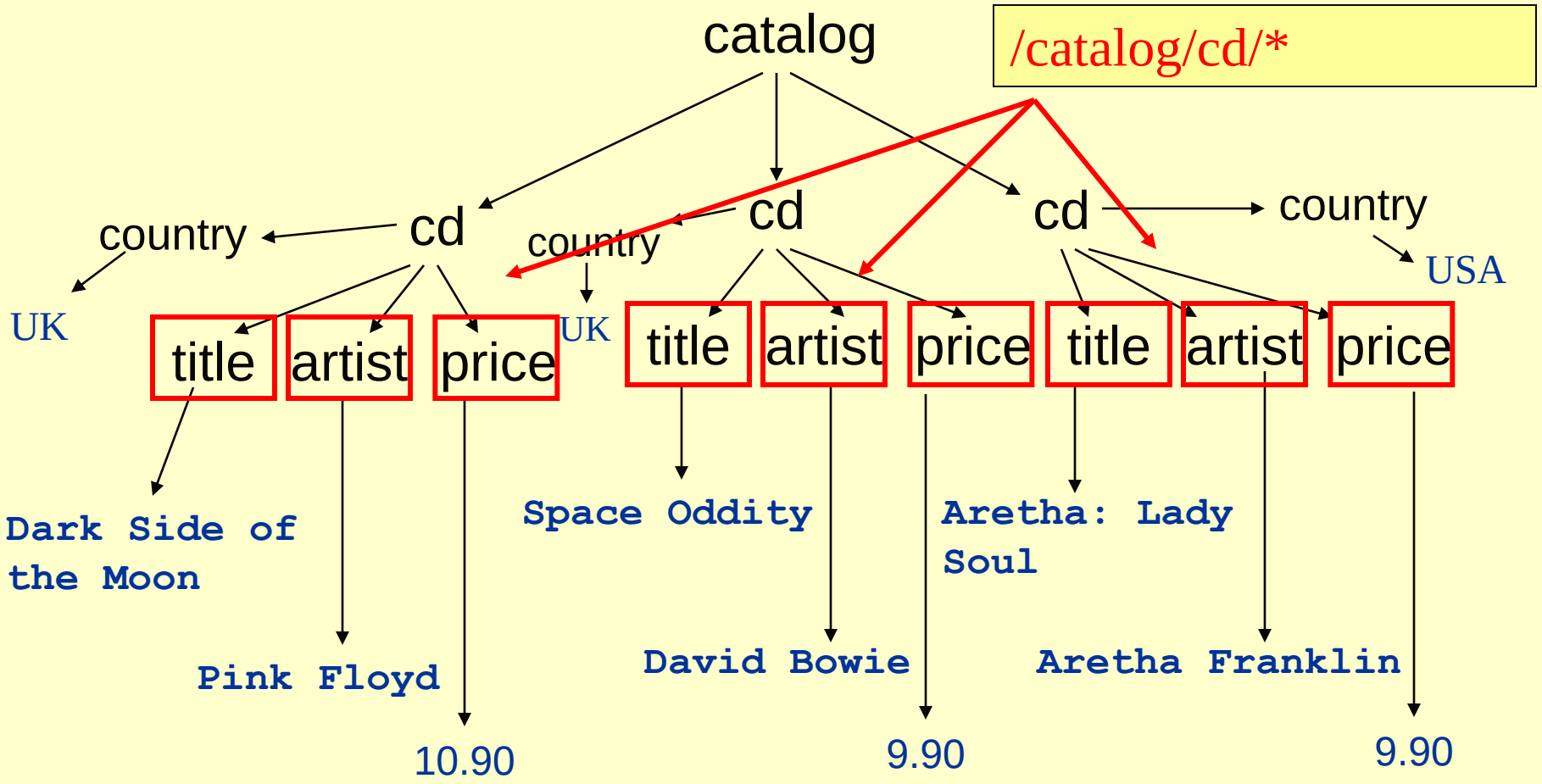


Condition on elements



// represents any directed path in the document

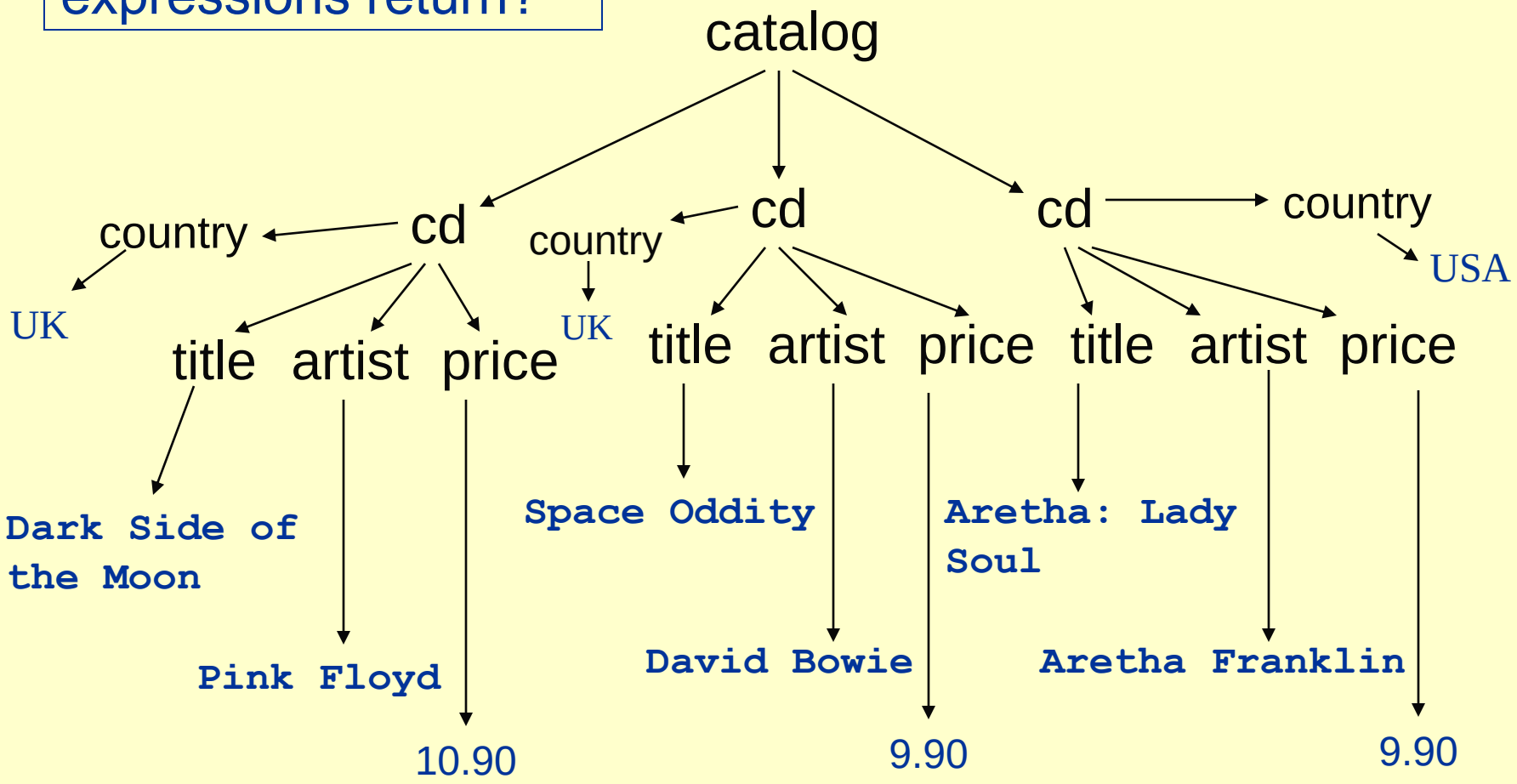
catalog.xml



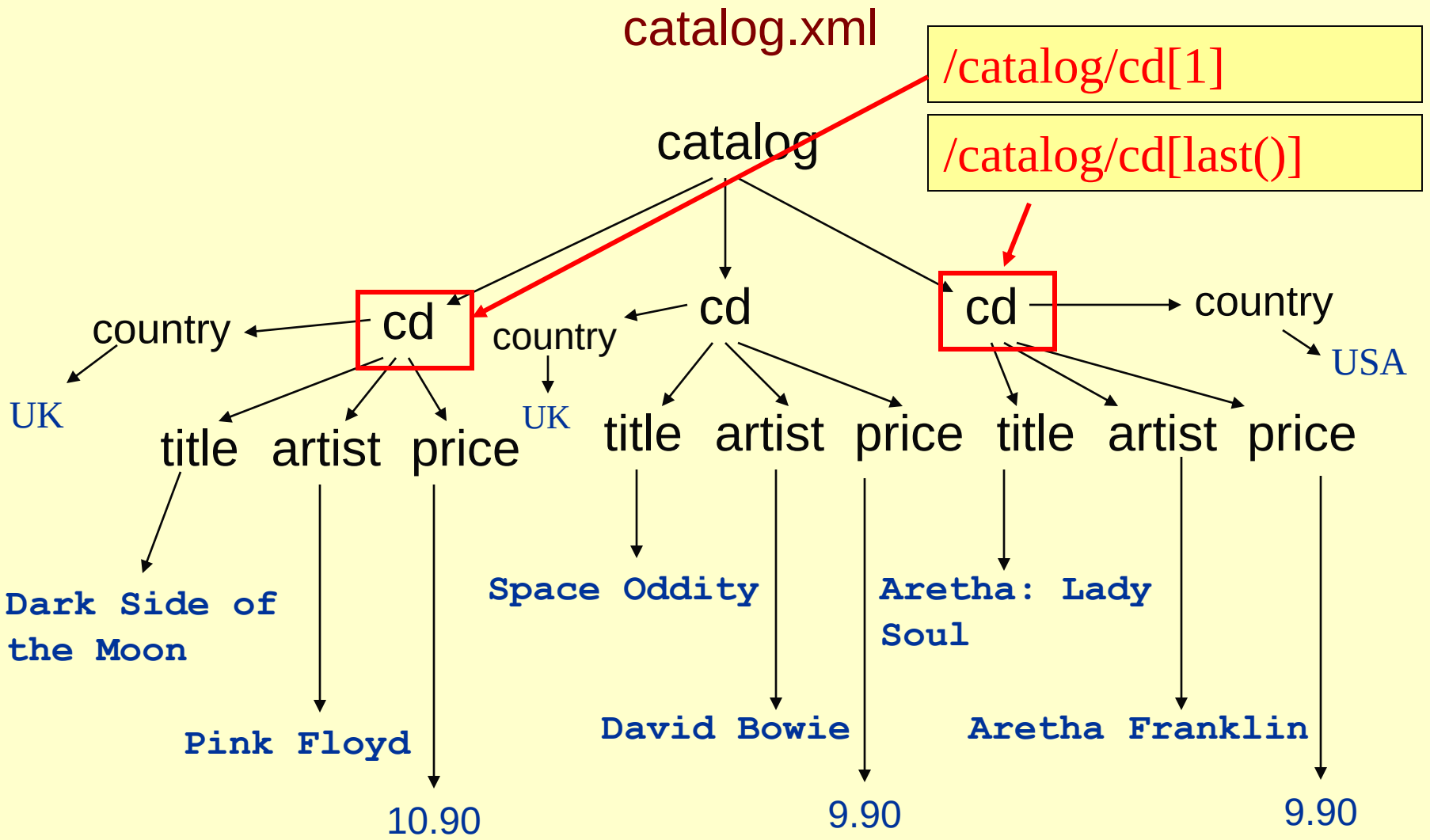
* represents any element name in the document

What will the following expressions return?

catalog.xml

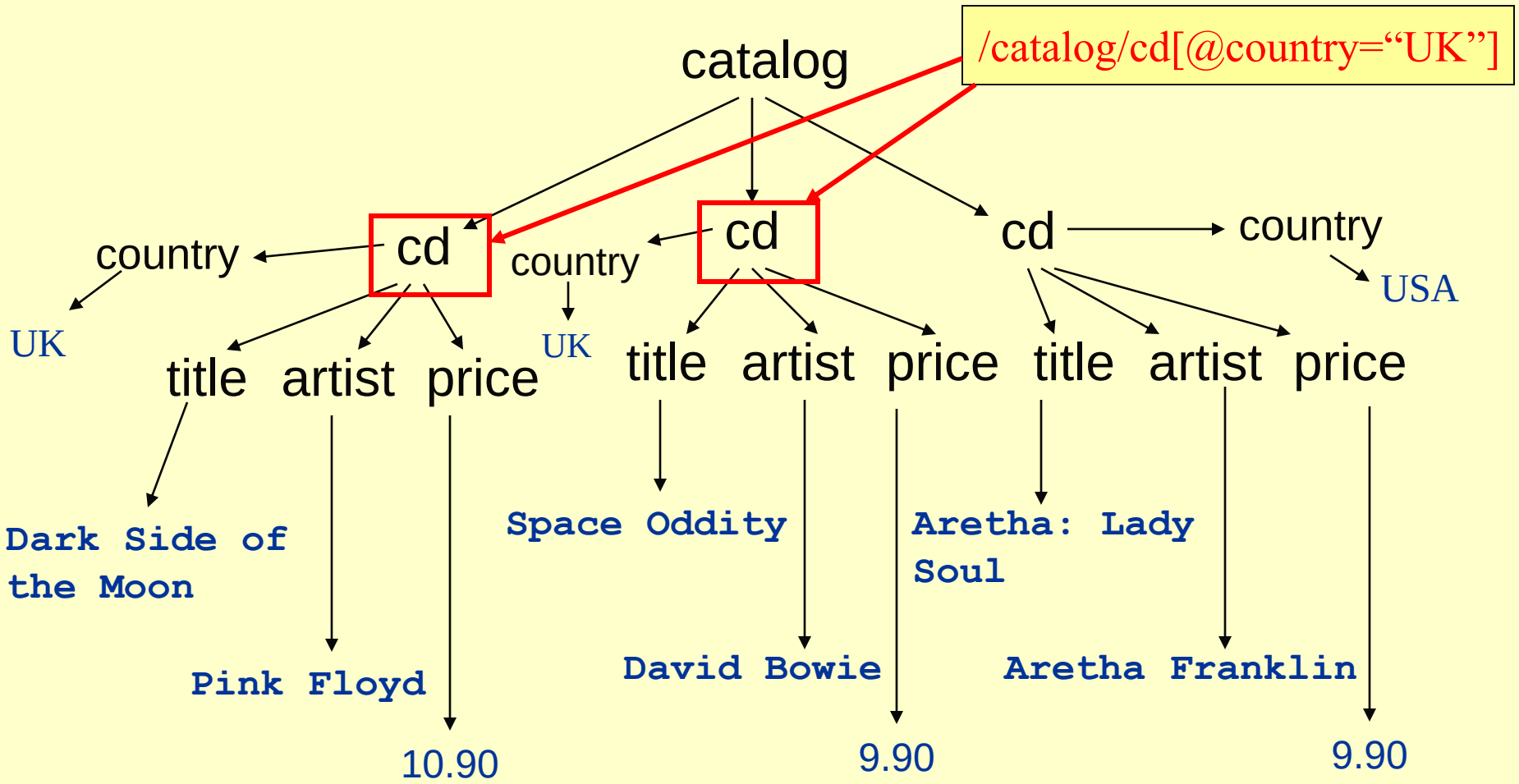


* represents any element name in the document



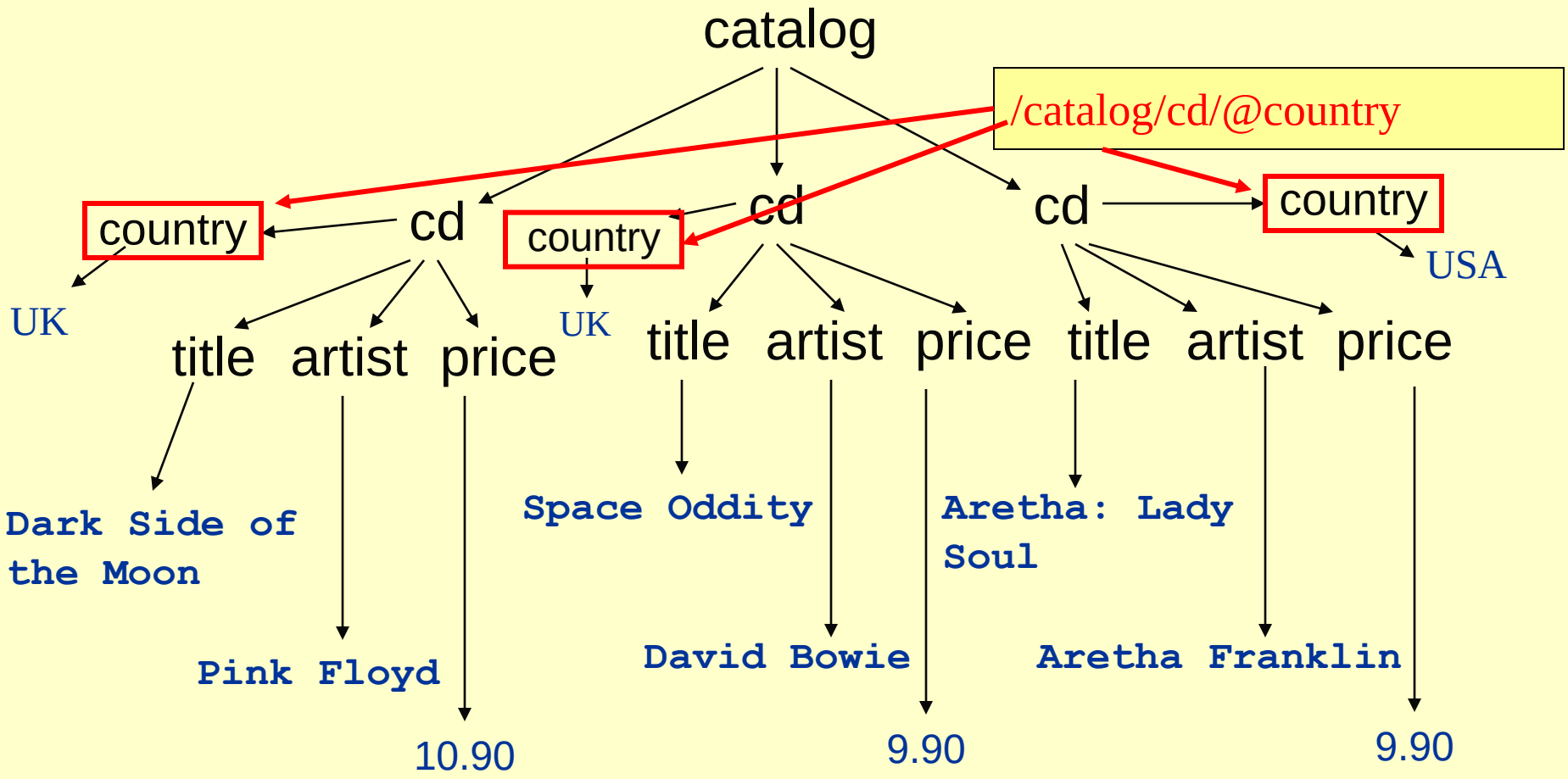
Position based condition

catalog.xml



@ marks attributes

catalog.xml



@ marks attributes

Some more Examples of XPath Expressions

- para
  - Selects the *para* children elements of the context node
- *
  - Selects all element children of the context node
- text()
  - Selects all text node children of the context node
- @name
  - Selects the *name* attribute of the context node

More Examples of XPath Expressions

- /doc/chapter[5]/section[2]
  - Selects the second *section* of the fifth *chapter* of the *doc*
- chapter//para
  - Selects the *para* element descendants of the *chapter* element children of the context node
- //para
  - Selects all the *para* descendants of the document root and thus selects all *para* elements in the same document as the context node

More Examples of XPath Expressions

- //olist/item
  - Selects all the *item* elements that have an *olist* parent and are in the same document as the context node
- .
  - Selects the context node
- .//para
  - Selects the *para* descendants of the context node
- ..
  - Selects the parent of the context node

More Examples of XPath Expressions

- ../@lang
  - Selects the *lang* attribute of the parent of the context node
- para[@type="warning"]
  - Selects the *para* children of the context node that have a *type* attribute with value *warning*
- chapter[title]
  - Selects the *chapter* children of the context node that have one or more *title* children

More Examples of XPath Expressions

- para[@type="warning"][5]
  - Selects the fifth *para* child among the children of the context node that have a *type* attribute with value *warning*

- para[5][@type="warning"]
  - Selects the fifth *para* child of the context node if that child has a *type* attribute with value *warning*

More Examples of XPath Expressions

- chapter[title="Introduction"]
  - Selects the *chapter* children of the context node that have one or more *title* children with string-value equal to *Introduction*

- employee[@secretary and @assistant]
  - Selects *employee* children of the context node that have both a *secretary* attribute and an *assistant* attribute

More Examples of Xpath Expressions

- */university/department/course*
  - This Xpath expression matches any path that starts at the root, which is a *university* element, passes through a *department* element and ends in a *course* element

- *./department/course[@year=2002]*
  - This Xpath expression matches any path that starts at the current element, continues to a child which is a *department* element and ends at a *course* element with a *year* attribute that is equal to 2002