XPath

XPath

- It provides a way to refer to specific parts of an XML tree
- An ‘URL- like’ scheme for locating documents on local and remote computer systems.
- A ‘foundation’ XML
  - related standard for linking and identifying XML resources, resemble to URL in locating Web resources.
- Used by other XML technologies
  - XSLT
  - Xpointer
  - Xquery Language

XPath Basics

- XPath uses a text string that contains meaningful symbols, called - “expression”.
- This linking scheme avoid problems when including navigation instructions in URLs.
- XPath provides a “foundation backbone” for other XML related standards such as XSLT, XPointer and XQL.
- The specification of XPath can be found in W3C link
  - http://www.w3.org/TR/xpath20/

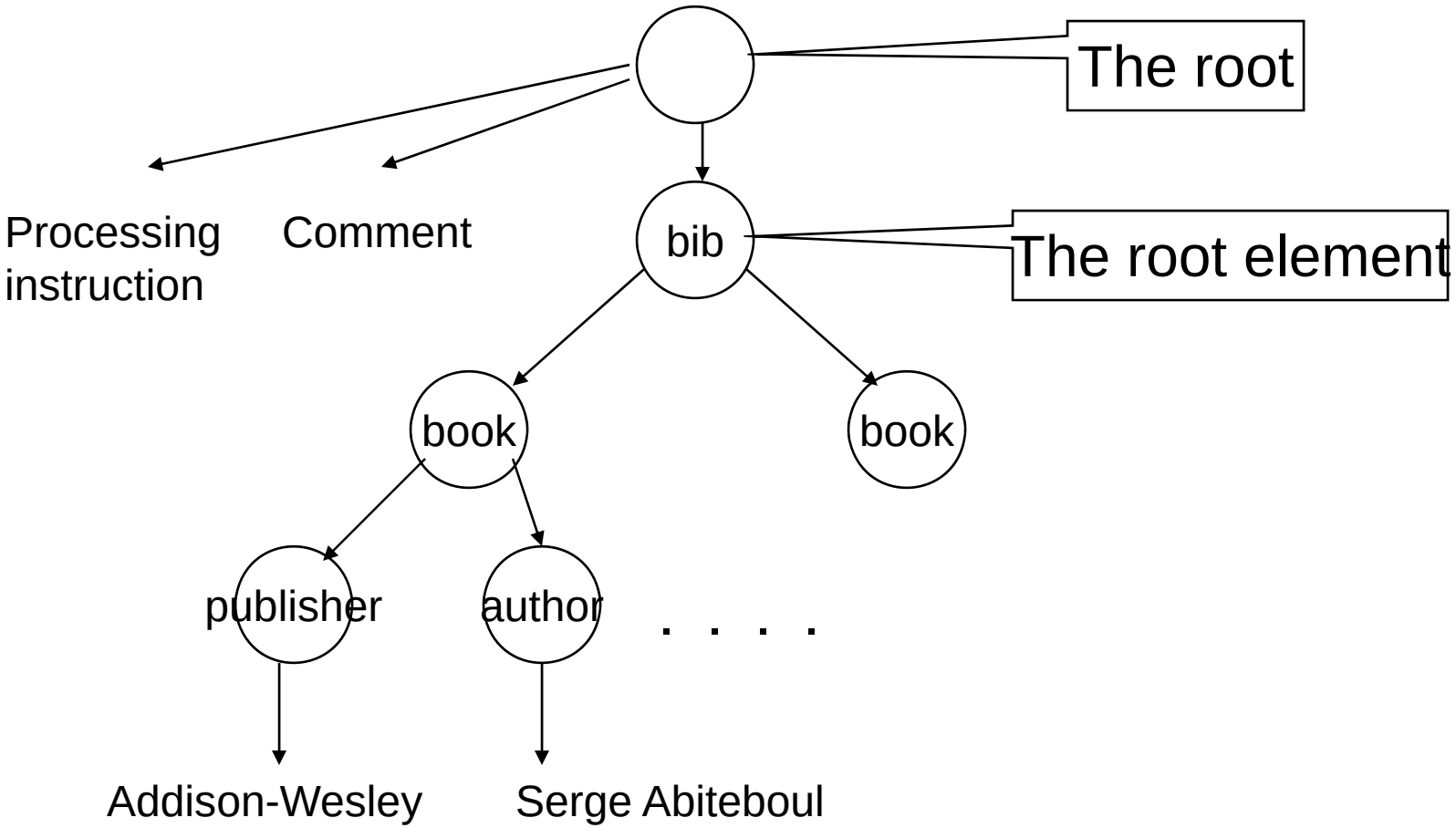
Nodes

- XML document
  - Tree structure with nodes
  - Each node represents part of XML document
    - Seven types
      - Root
      - Element
      - Attribute
      - Text
      - Comment
      - Processing instruction
      - Namespace
    - Attributes and namespaces are not children of their parent node
      - They describe their parent node

Example for XPath Queries

```
<bib>
    <book>  <publisher> Addison-Wesley </publisher>
            <author> Serge Abiteboul </author>
            <author> <first-name> Rick </first-name>
                     <last-name> Hull </last-name>
            </author>
            <author> Victor Vianu </author>
            <title> Foundations of Databases </title>
            <year> 1995 </year>
    </book>
    <book price="55">
            <publisher> Freeman </publisher>
            <author> Jeffrey D. Ullman </author>
            <title> Principles of Database and Knowledge Base Systems </title>
            <year> 1998 </year>
    </book>
</bib>
```

Data Model for XPath



Relative Path

- A Relative Path is one that starts from an existing location in the document
- The simplest form is a single path name:
  - e. g. author to link- up a XML element in the current context
- child path (child ::)
  - e. g. child:: author (note: child doesn't mean child of "author", but rather the child of current element "author")
- Wildcards (*)
  - e. g. book/*/ firstname (note: uses it with care, or …)
- parent path ("..” / " parent:: ”)
  - e. g. ../ title or parent:: node()/ child:: title
- Grandparent
  - e. g. parent:: node()/ parent:: node()/ child:: title

Relative Paths

- A relative location path has the following syntax:

  Path ::= $Step_1/\ldots/Step_n$

  where each *Step* is a triple (*Axis*, *Node-test*, *Predicate*) and is defined as follows:

  Step ::= Axis:: Node-test Predicate*

  – The axis specifies the direction to move in the document tree

  – The node test selects nodes along the specified axis, and

  – The predicates (if any) filter the nodes selected

- Separators “/” between two subsequent steps indicate a parent - child relationship between nodes

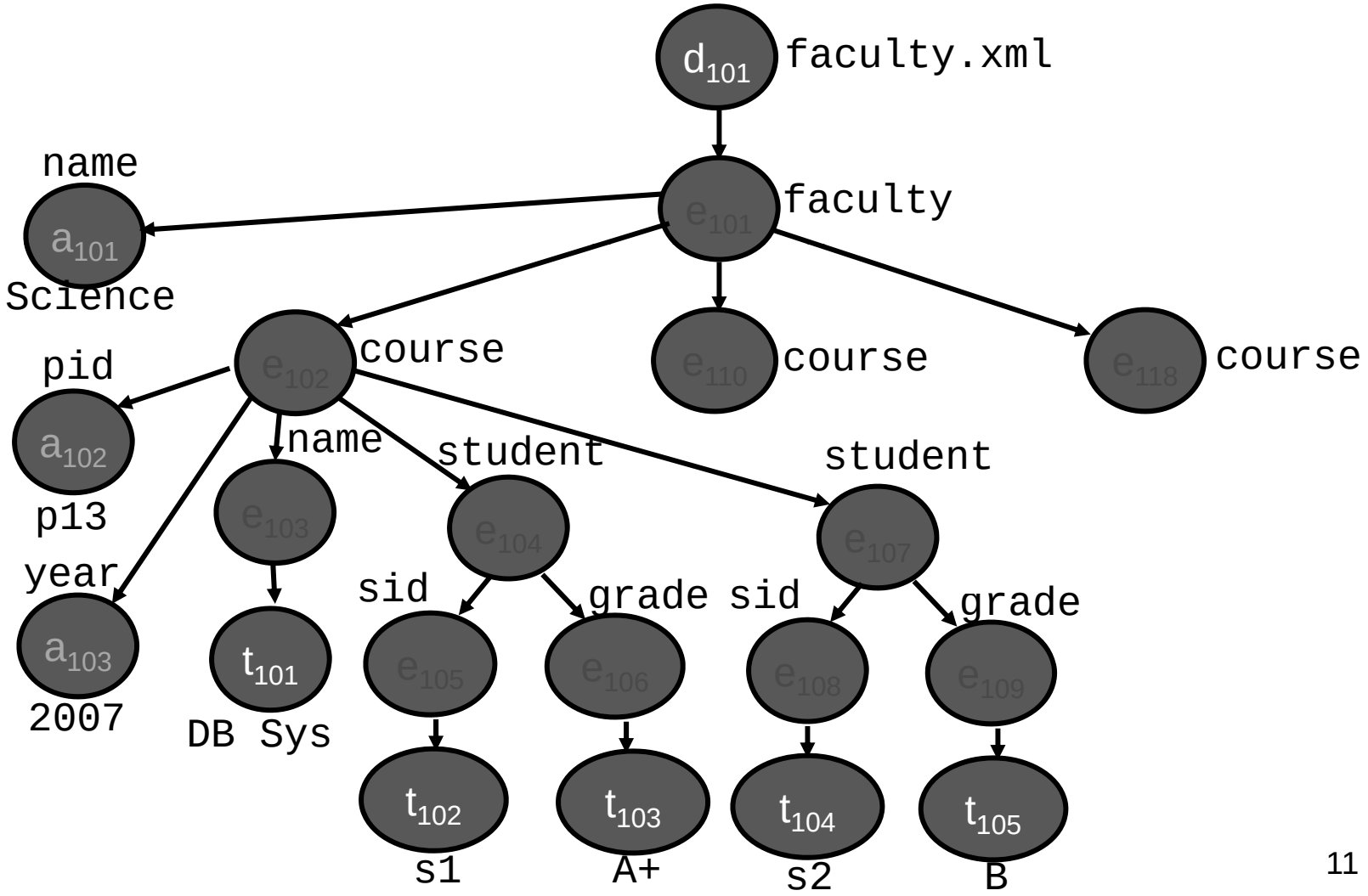
Evaluation of a Relative Location Path

- A relative location path is evaluated step by step, from left to right
- A step is applied to a single node, so called ***context*** node
- The application of a step on a context node selects a sequence of result nodes
- Each node of a result sequence is then used as a context node in the following step
- The **result** of an expression is a concatenation of node-sequences selected by the **last** step

Absolute Path

- For some cases, a relative path is not suitable.
  - E.g. it may be necessary to select the title of a book, regardless of the current context. In which the location relative to the document (as a whole) may be know, whereas the offset from the current location may not - use absolute path.
- An absolute path is similar to relative path, except for the front part start with “/” - root of document
  - e. g. / book/ title
- Use “//” expression, it can even possible to select all occurrences of a specific element type.
  - e. g. // author
- Use a specific “anchor point” to create absolute path.
  - e. g. id(“ pic123”)...

Partial Tree of `faculty.xml`



What Does an XPath Expression Return?

- A sequence of result nodes with their contents in the form of an (not necessarily well formed) XML document
- XPath:

  doc("faculty.xml")/descendant::course[attribute::pid="p13"]/child::student[child::sid="s2"]

- Result:

```
<student>
   <sid>s2</sid>
   <grade>B</grade>
</student>
<student>
  <sid>s2</sid>
  <grade>D</grade>
</student>
```

Example for XPath Queries

```
<bib>
    <book>  <publisher> Addison-Wesley </publisher>
            <author> Serge Abiteboul </author>
            <author> <first-name> Rick </first-name>
                     <last-name> Hull </last-name>
            </author>
            <author> Victor Vianu </author>
            <title> Foundations of Databases </title>
            <year> 1995 </year>
    </book>
    <book price="55">
            <publisher> Freeman </publisher>
            <author> Jeffrey D. Ullman </author>
            <title> Principles of Database and Knowledge Base Systems </title>
            <year> 1998 </year>
    </book>
</bib>
```

XPath: Simple Expressions

/bib/book/year

Result: <year> 1995 </year>
<year> 1998 </year>

/bib/paper/year

Result: empty (there were no papers)

XPath: //

//author

Result:<author> Serge Abiteboul </author>
<author> <first-name> Rick </first-name>
<last-name> Hull </last-name>
</author>
<author> Victor Vianu </author>
<author> Jeffrey D. Ullman </author>

/bib//first-name

Result: <first-name> Rick </first-name>

Xpath: Functions

/bib/book/author/text()

Result: Serge Abiteboul
Victor Vianu
Jeffrey D. Ullman

Rick Hull doesn't appear because he has firstname, lastname

Functions in XPath:

- text() = matches the text value
- node() = matches any node (= * or @* or text())
- name() = returns the name of the current tag

• http://www.w3.org/TR/xquery-operators/

Xpath: Wildcard

//author/*

Result: <first-name> Rick </first-name>
<last-name> Hull </last-name>

* Matches any element

Xpath: Attribute Nodes

/bib/book/@price

Result: “55”

@price means that price is has to be an attribute

Xpath: Qualifiers

/bib/book/author[firstname]

Result: <author> <first-name> Rick </first-name>
<last-name> Hull </last-name>
</author>

Xpath: More Qualifiers

/bib/book/author[firstname][address[//zip][city]]/lastname

Result: <lastname> … </lastname>

<lastname> … </lastname>

Xpath: More Qualifiers

/bib/book[@price < “60”]

/bib/book[author/@age < “25”]

/bib/book[author/text()]

Axes

- XPath searches are made relative to *context node*
- Axis
  - Indicates which nodes are included in search
    - Relative to context node
  - Dictates node ordering in set
    - Forward axes select nodes that follow context node
    - Reverse axes select nodes that precede context node

XPath axes

Axis Name	Ordering	Description
`self`	none	The context node itself.
`parent`	reverse	The context node's parent, if one exists.
`child`	forward	The context node's children, if they exist.
`ancestor`	reverse	The context node's ancestors, if they exist.
`ancestor-or-self`	reverse	The context node's ancestors and also itself.
`descendant`	forward	The context node's descendants.
`descendant-or-self`	forward	The context node's descendants and also itself.
`following`	forward	The nodes in the XML document following the context node, not including descendants.
`following-sibling`	forward	The sibling nodes following the context node.
`preceding`	reverse	The nodes in the XML document preceding the context node, not including ancestors.
`preceding-sibling`	reverse	The sibling nodes preceding the context node.
`attribute`	forward	The attribute nodes of the context node.
`namespace`	forward	The namespace nodes of the context node.

Node Tests

- Node tests
  - Refine set of nodes selected by axis
    - Rely upon axis' *principle node type*
      - Corresponds to type of node axis can select

Node Test	Description
`*`	Selects all nodes of the same principal node type.
`node()`	Selects all nodes, regardless of their type.
`text()`	Selects all text nodes.
`comment()`	Selects all comment nodes.
`processing-instruction()`	Selects all processing-instruction nodes.
node name	Selects all nodes with the specified *node name*.

```
1  <?xml version = "1.0"?>
2
3  <!-- Fig. 11.9 : books.xml -->
4  <!-- XML book list         -->
5
6  <books>
7
8     <book>
9        <title>Java How to Program</title>
10       <translation edition = "1">Spanish</translation>
11       <translation edition = "1">Chinese</translation>
12       <translation edition = "1">Japanese</translation>
13       <translation edition = "2">French</translation>
14       <translation edition = "2">Japanese</translation>
15    </book>
16
17    <book>
18       <title>C++ How to Program</title>
19       <translation edition = "1">Korean</translation>
20       <translation edition = "2">French</translation>
21       <translation edition = "2">Spanish</translation>
22       <translation edition = "3">Italian</translation>
23       <translation edition = "3">Japanese</translation>
24    </book>
25
26 </books>
```

Location Paths Using Axes and Node Tests

- Which books have Japanese translations?
  - Use root node of XPath tree as context node
  - Use predicate
    - Boolean expression for filtering nodes from search
    - Compare string value of current node to string '**`Japanese`**'

```
/books/book/translation[. = 'Japanese']/../title
```

Node-set Operators and Functions

- Node-set operators
  - Manipulate node sets to form others
- Node-set functions
  - Perform actions on node-sets returned by location paths

Node-set Operators	Description
pipe (`|`)	Performs the union of two node-sets.
slash (`/`)	Separates location steps.
double-slash (`//`)	Abbreviation for the location path `/descendant-or-self::node()/`

Some node-set functions

Node-set Functions	Description
`last()`	Returns the number of nodes in the node-set.
`position()`	Returns the position number of the current node in the node-set being tested.
`count(` *node-set* **`)`**	Returns the number of nodes in *node-set*.
`id(` *string* **`)`**	Returns the element node whose ID attribute matches the value specified by argument *string*.
`local-name(` *node-set* **`)`**	Returns the local part of the expanded-name for the first node in *node-set*.
`namespace-uri(` *node-set* **`)`**	Returns the namespace URI of the expanded-name for the first node in *node-set*.
`name(` *node-set* **`)`**	Returns the qualified name for the first node in *node-set*.

Node-set Operators and Functions

- Location-path expressions
  - Combine node-set operators and functions
    - Select all **`head`** and **`body`** children element nodes
      `head | body`
    - Select last **`bold`** element node in **`head`** element node
      `head/title[ last() ]`
    - Select third book element
      `book[ position() = 3 ]`
      - Or alternatively
        `book[ 3 ]`
    - Return total number of element-node children
      `count( * )`
    - Select all book element nodes in document
      `//book`