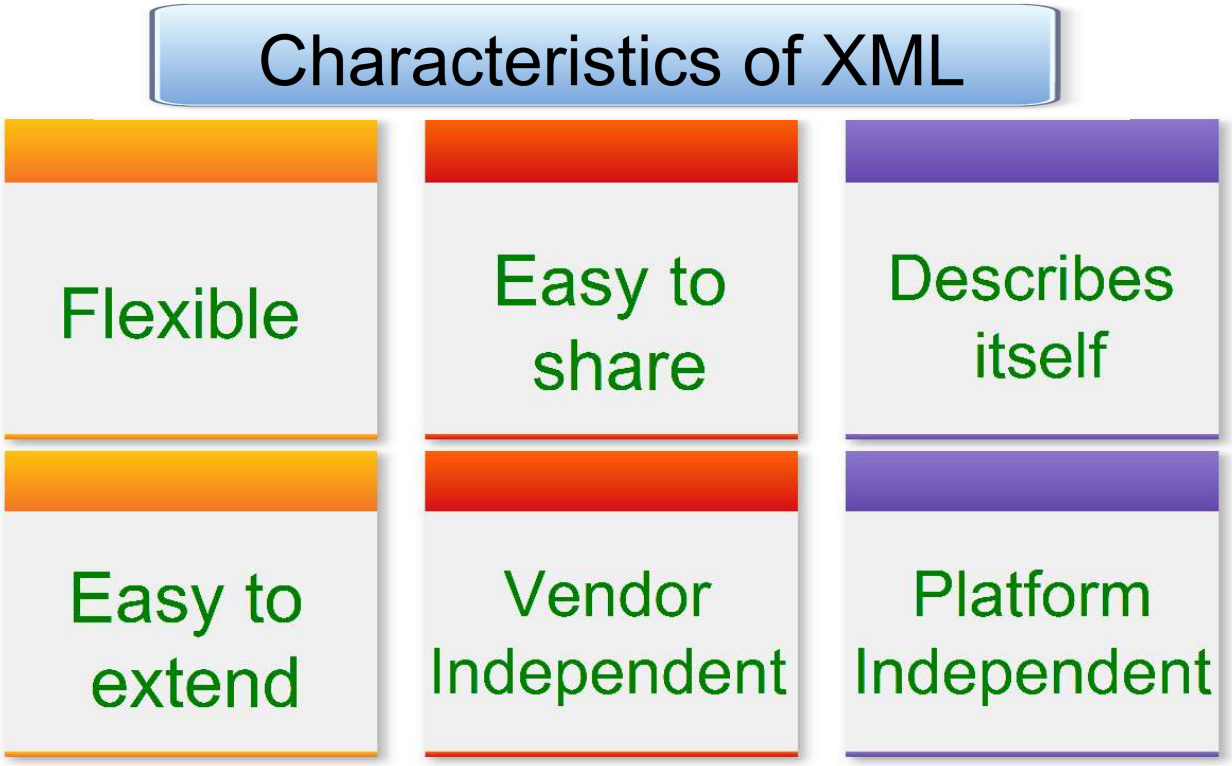
DB2 pureXML

IBM Information Management Cloud Computing Center of Competence
IBM Canada Lab

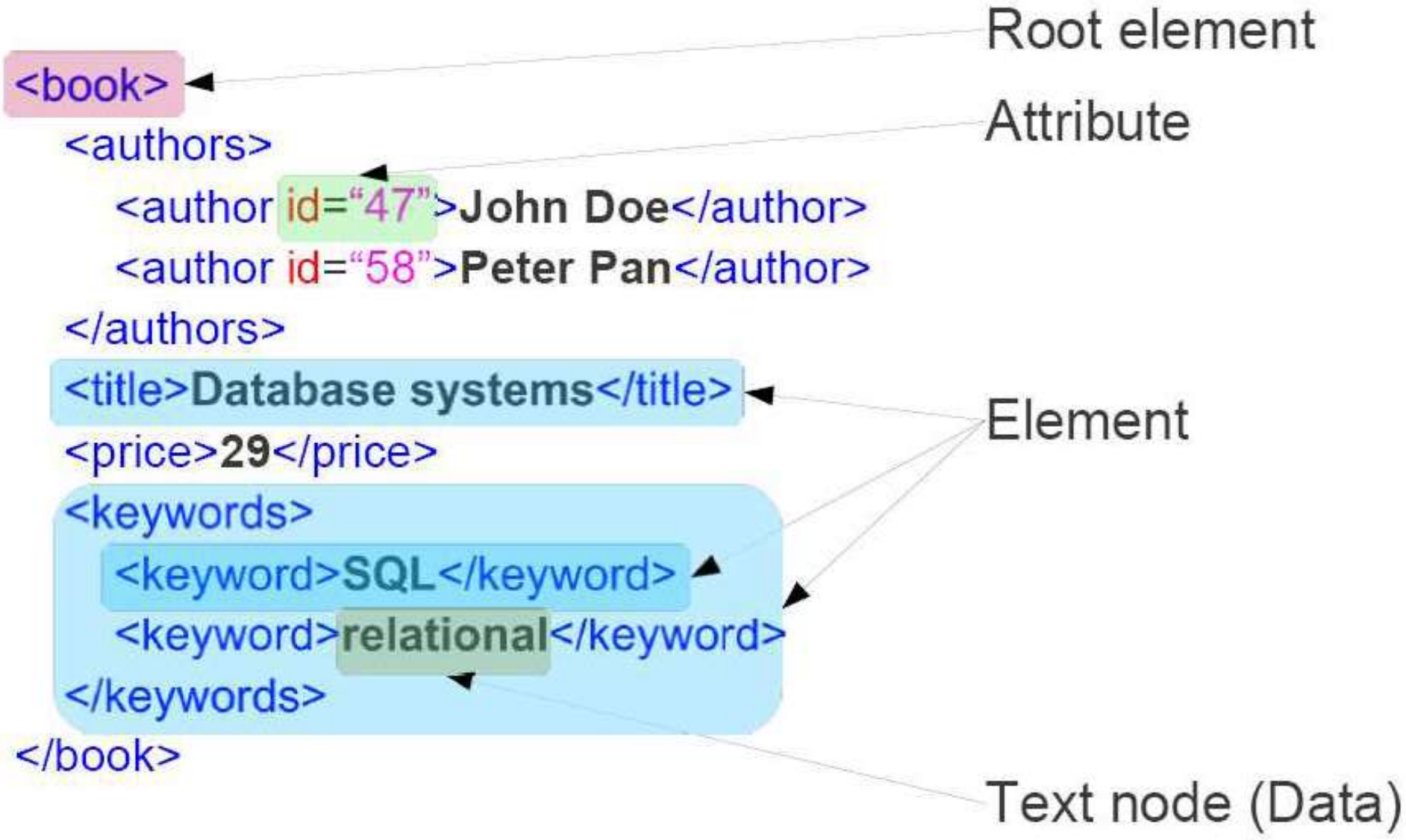
What is XML?

- **e*X*tensible *M*arkup *L*anguage**
  - –XML is a language designed to describe data
- **A hierarchical data model**

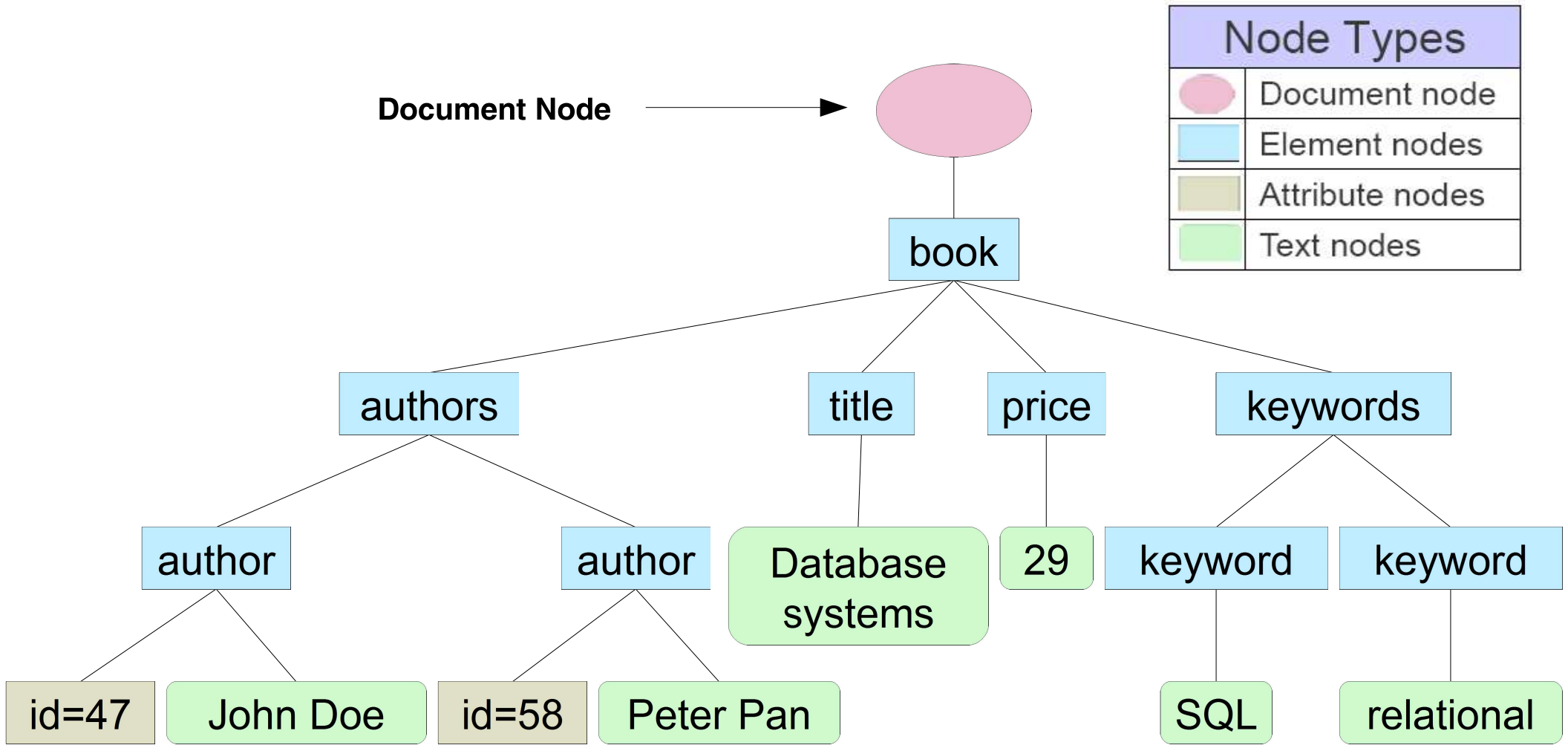
```
<book>
    <authors>
        <author  id="47">John Doe</author>
        <author  id="58">Peter Pan</author>
    </authors>
    <title>Database systems</title>
</book>
```



XML document: Serialized representation



XML document: Parsed-hierarchical representation



Well-formed vs. valid XML documents

- **A well-formed XML document is a document that follows basic rules:**
  1) It must have one and only one root element
  2) Each element begins with a start tag and ends with an end tag
  3) An element can contain other elements, attributes, or text nodes
  4) Attribute values must be enclosed in double quotes. Text nodes, on the other hand, should not.

- **A valid XML document is BOTH:**
  1) A well-formed XML document
  2) A document compliant with the rules defined in an XML schema document or a Document Type Definition (DTD) document.

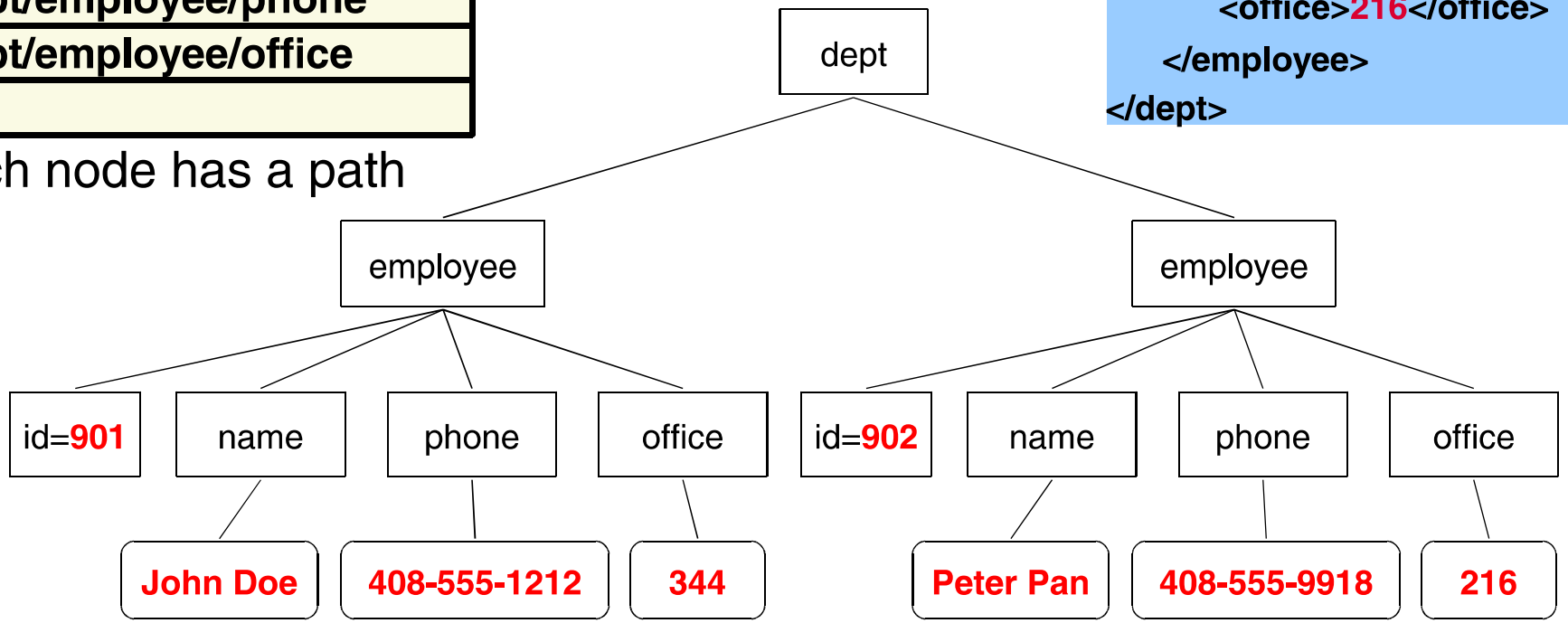
XPath

- XML Query Language
- Subset of XQuery & SQL/XML

/
/dept
/dept/employee
/dept/employee/@id
/dept/employee/name
/dept/employee/phone
/dept/employee/office
(...)

Each node has a path

```
<dept bldg="101">
  <employee id="901">
    <name>John Doe</name>
      <phone>408 555 1212</phone>
      <office>344</office>
  </employee>
  <employee id="902">
      <name>Peter Pan</name>
      <phone>408 555 9918</phone>
      <office>216</office>
  </employee>
</dept>
```



XPath: Simple expressions

- Use fully qualified paths to specify elements/attributes
- "@" is used to specify an attribute
- use "text()" to specify the text node under an element

```
<dept bldg="101">
  <employee id="901">
    <name>John Doe</name>
      <phone>408 555 1212</phone>
      <office>344</office>
  </employee>
  <employee id="902">
      <name>Peter Pan</name>
      <phone>408 555 9918</phone>
      <office>216</office>
  </employee>
</dept>
```

XPath	Result
/dept/@bldg	101
/dept/employee/@id	901 902
/dept/employee/name	<name>John Doe</name> <name>Peter Pan</name>
/dept/employee/name/text()	Peter Pan John Doe

XPath: Wildcards

- * matches any tag name
- // is the “descendent-or-self” wildcard

```
<dept bldg="101">
  <employee id="901">
    <name>John Doe</name>
      <phone>408 555 1212</phone>
      <office>344</office>
  </employee>
  <employee id="902">
      <name>Peter Pan</name>
      <phone>408 555 9918</phone>
      <office>216</office>
  </employee>
</dept>
```

XPath	Result
/dept/employee/*/text()	John Doe 408 555 1212 344 Peter Pan 408 555 9918 216
/dept/*/@id	901 902
//name/text()	John Doe Peter Pan
/dept//phone	<phone>408 555 1212</phone> <phone>408 555 9918</phone>

XPath: Predicates

- Predicates are enclosed in square brackets […]
- Can have multiple predicates in one XPath
- Positional predicates: [n] selects the n-th child

```
<dept bldg="101">
  <employee id="901">
    <name>John Doe</name>
      <phone>408 555 1212</phone>
      <office>344</office>
  </employee>
  <employee id="902">
      <name>Peter Pan</name>
      <phone>408 555 9918</phone>
      <office>216</office>
  </employee>
</dept>
```

XPath	Result
/dept/employee[@id="902"]/name	<name>Peter Pan</name>
/dept[@bldg="101"]/employee[office >"300"]/name	<name>John Doe</name>
//employee[office="344" OR office="216"]/@id	901 902
/dept/employee[2]/@id	902

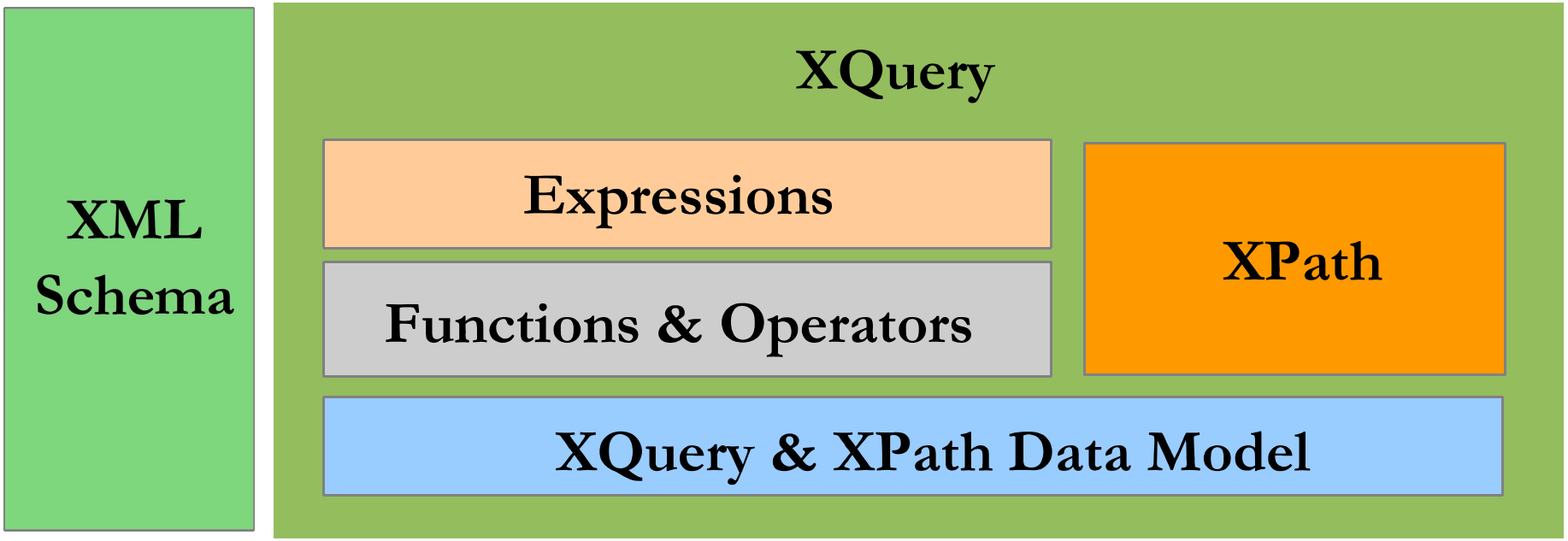
XPath: Parent axis

- Current context: “.”
- Parent context: “..”

```
<dept bldg=“101”>
  <employee id=“901”>
    <name>John Doe</name>
      <phone>408 555 1212</phone>
      <office>344</office>
  </employee>
  <employee id=“902”>
      <name>Peter Pan</name>
      <phone>408 555 9918</phone>
      <office>216</office>
  </employee>
</dept>
```

XPath	Result
/dept/employee/name[../@id=“902”]	<name>Peter Pan</name>
/dept/employee/office[.>“300”]	<office>344</office>
/dept/employee[office > “300”]/office	<office>344</office>
/dept/employee[name=“John Doe”]/../@bldg	101
/dept/employee/name[.=“John Doe”]/../../@bldg	101

What is XQuery?



- XQuery supports path expressions to navigate XML
- XQuery supports both typed and untyped data
- XQuery lacks null values because XML documents omit missing or unknown data
- XQuery returns sequences of XML data

XQuery: The FLWOR expression

- **FOR:** iterates through a sequence, bind variable to items
- **LET:** binds a variable to a sequence
- **WHERE:** eliminates items of the iteration
- **ORDER:** reorders items of the iteration
- **RETURN:** constructs query results