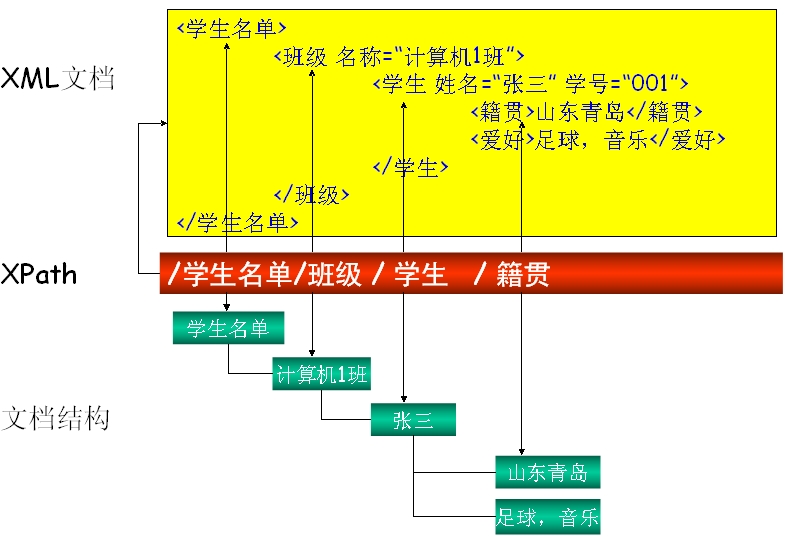
**Xpath**

**概述**

XPath在设计之初主要用于XSLT和XPointer(用于Xlink，还未普及)，随着XSLT 2.0的发布，已经发展到XPath 2.0（06年6月），并成为XSLT 2.0和XQuery 2.0的基础  
XPath以“路径”方式查询XML文档，XPath表达式的基本形式是“/结点/子结点/二级子结点”，从左到右（即从外至内）匹配XML文档的结点  
XPath表达式分为定位表达式和求值表达式。定位表达式用于匹配XML文档的结点，而求值表达式则返回定位结果数值（字符串、布尔值、数值等）  
**XML文档的模型**对于良构的XML文档，有三种模型表示：  
XPath数据模型：把XML文档的多数内容表示为一棵结点树，树的根结点代表文档本身，其他结点有元素结点、属性结点、文本结点、名称空间结点、注释结点等。XML声明、DOCTYPE等不能表示为结点  
DOM：采用树形层次结构表示XML文档  
XML信息表（Infoset）：把XML文档看作由信息项组成的一棵树，每个信息项相当于XPath的一个结点，信息项可有若干属性，属性值表示信息项的特性



**基本概念  
层次结构** XPath表达式以“/”分隔表示分层结构  
XPath表达式最左边的“/”表示从根结点开始  
XPath表达式中间出现的“/”用于分隔上下两层结构，表示父子关系  
串联的若干“/”分隔的部分形成先代（Ancestor）、后代（Descendant）关系

**线性结构**  
 同属一个父结点的结点间形成线性的兄弟（Sibling）关系  
兄弟之间有先后，前面的是后面的“前导（Preceding）兄弟”后面的是前面的“后继（Following）兄弟”  
可以用position()函数获取结点在线性结构中的位置

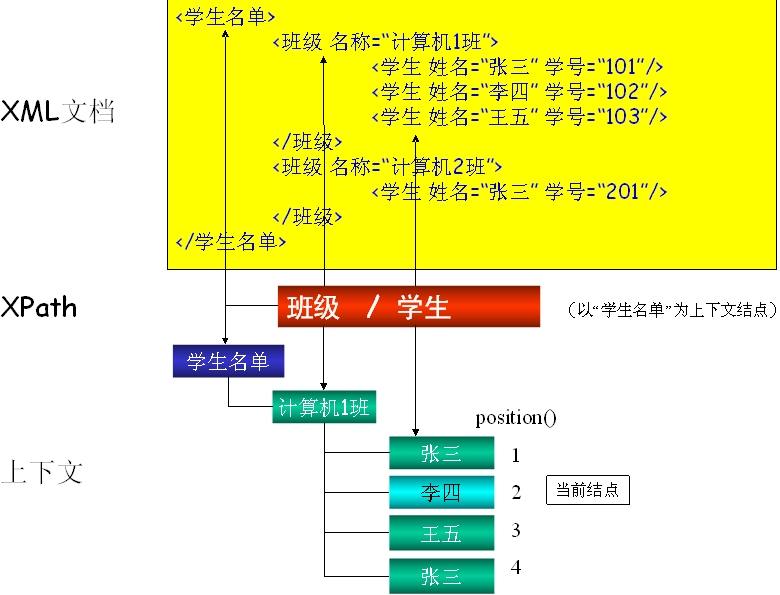
XPath定位操作返回的是结点列表  
 XPath定位表达式返回的是匹配的结点的列表，不是返回值  
 XPath求值表达式可以返回结点集

根结点不是根元素  
 XPath的根结点与XML文档结点对应，包含了文档的所有内容，不是XML文档的根元素，根元素是XML文档包含其它元素的结点，是根结点的一个子结点（根结点的其它子结点包括XML声明、 DOCTYPE、注释等）

绝对定位和相对定位  
 若XPath表达式以“/”开始，则对应为绝对定位；否则为相对定位

**XPath的概念**

XPath路径的构造方法  
 由于XML文档是层次结构的，定位元素不可能像平面结构的关系数据库可以根据主键一步到位，只能是分步渐进的，每个分步称为一个“定位步（location step）”  
 XPath的路径有两种：  
串联定位：使用“/”分隔符连接多个定位步  
 如：/我的电脑/硬盘/分区  
 先定位根结点，然后是“我的电脑”，接着是“硬盘”，最后是“分区”  
并联定位：使用“|”分隔符连接多个定位步  
 如：硬盘 | 内存  
 在上下文结点的子结点中定位所有“硬盘”和“内存”结点  
XPath表达式的上下文  
在某结点下的XPath表达式的计算结果——结点列表构成上下文，该结点称为上下文结点（Context node）  
XPath表达式的结点列表中包含的结点的数目称为该上下文的大小（Context Size）  
在遍历XPath的结点列表时，遍历游标所到达的结点，称为当前结点



定位步的组成

XPath的定位步由三部分组成，格式为：

轴::结点测试[谓项]

其中“轴”和“谓项”部分可以省略  
轴（Axis）：表示从当前结点开始，向哪个方向开后匹配结点。XPath定义了13条轴  
结点测试：表示匹配轴方向上的哪些结点  
谓项（Predicate）：表示匹配的条件，是在轴和结点测试基础上的进一步筛选

**轴和结点测试**

**13条轴**

  
**结点测试** 在XPath中，结点测试必须位于轴的后面或放在谓项中，而函数的返回值只能出现在谓项中  
结点测试：不支持DTD的DOCTYPE、ENTITY等  
 元素名称 ： 具有特定名称的结点  
名称空间前缀:元素名称 ： 名称空间  
processing-instruction(“名称") ： 处理指令  
node() ： 任何类型的结点  
\* ： 现节点下所有元素，具有任何名称的结点  
\*/elem ： 现节点下所有节点的子节点中为elem的节点  
@Prop ： 属性值  
@\* ： 所有现节点的属性  
. ： 现节点  
.. ： 现节点上级  
elem[i] ： 现节点下第i个叫做elem的元素  
elem[posetion()=1] ： 同上  
elem/[@prop="somevalue"] ： 现节点下，名字为elem，具有prop的属性，并且属性值为somevalue的那个元素  
elem1|elem2 ： 现节点下，名字为elem1或elem2的元素  
.//elem ： 现节点下，可以跨越级别，所有的名字叫做elem的元素  
elem1//elem2 ： 现节点下，可以跨越级别，所有的名字叫做elem2，且elem2的上级中有人叫elem1，且elem1是现节点的子元素的元素  
text() ： 现节点的子元素中所有的文字节点  
comment() ： 注释  
**函数：不能直接出现在轴的后边或结点测试的后边**contains(串1,串2) ：包含检查。如果 string1 包含 string2，则返回 true，否则返回 false。  
id(结点集等) ：返回具有id属性的结点  
name() ：结点名称  
position() ：当前结点在上下文中的位置  
starts-with(串1,串2) ：“串1”以“串2”开始  
count() ： count(PERSON[name='tom']) 计数  
number() ： select="number(book/price)" 转成数字  
substring(value,start,length) ： select="substring(name,1,3)"  
sum() ： select="sum(//price)" 求和

**谓项**

谓项（Predicate）以一对方括号“[“和”]”为标记，中间为条件表达式，置于XPath中，构成筛选条件，格式为：  
   [值1 操作符 值2]  
  若谓项条件不成立，则从XPath结点列表中剔除该结点  
**确定结点是否存在**谓项的格式为：[结点测试]  
例如：[姓名] 、[@姓名]  
**用关系表达式表示筛选条件**谓项格式为：[值1 关系运算符  值2]  
例如：[年龄>=“20”] 、 [@性别!=“男”]  
关系元素符包括：=、!=、<、<=、>=、>  
**用布尔运算表示复合条件**  
谓项格式为：[布尔值1  布尔运算符 布尔值2]  
例如：[年龄=“20” and @性别=“女”]  
布尔运算符包括：or（或）、and（与）  
**用数值表达式构造复杂运算**数值运算符包括： +、-、\*、div（除）、mod（求模）、-（取反）

例如：

1. **<?xml** version="1.0" encoding="utf-8"**?>**
2. **<**学生名单**>**
3. **<**班级 名称="计算机1班"**>**
4. **<**学生 姓名="张三" 学号="101"**/>**
5. **<**学生 姓名="李四" 学号="102"**/>**
6. **<**学生 姓名="王五" 学号="103"**/>**
7. **<**班长 学号="103"**>**
8. **<**电话 号码="13512345678"**/>**
9. **</**班长**>**
10. **<**介绍**>**该班为优良学风班**</**介绍**>**
11. **</**班级**>**
12. **<**班级 名称="计算机2班"**>**
13. **<**学生 姓名="张三" 学号="201"**/>**
14. **</**班级**>**
15. **</**学生名单**>**

实例1：假设上下文结点为“/学生名单/班级”，XPath匹配结果分别为：  
学生                     所有学生结点  
班长/电话            所有电话结点  
..                           班级结点  
../..                        学生名单结点  
./介绍                   介绍结点  
//学生                   所有学生结点  
self::\*//学生         当前结点下含有学生子结点的任意结点  
/descendant::node() 根节点的所有后代结点  
//node()               所有节点  
.//node()              当前结点下的所有结点  
“介绍”文本     介绍/text()  
每个“学生”的“姓名”学生/@姓名

参照实例（1），假设上下文结点为“学生名单”，写出路径的XPath：  
有“班长”的所有班级                              班级[班长]  
包含“电话”子结点的所有结点              班级/\*[电话]  
叫“张三”的“学生”                              班级/学生[@姓名="张三"]  
“计算机1班”叫“张三”的“学生”    班级[@名称="计算机1班"]/学生[@姓名="张三"]

**名称空间**

结点的本地名称可以用函数local-name()获取 例如：  //\*[local-name()=“学生”]  
结点的名称空间可用函数namespace-uri()获取 例如：  //\*[namespace-uri()=“myurn:computer”]  
在XPath中使用名称空间前缀限定名称空间 例如：  //学籍管理:学生[@学籍管理:姓名=“张三”]