China-pub.com
下载

第5章　使用XDR模式创建XML视图

回顾第4章，你会发现那一章包含大量的信息和编码技术。这些信息使我们可以通过以下几种方式创建XML文档：调用存储过程、调用模板文件、在URL中直接声明SQL语句。模板文件的内容也可以直接列在URL中。另外，通过应用XSLT样式表，我们可以修改返回的文档以适合我们的需求。

即使我们已经学习了如何完成这些工作，我们还需要学习XPath表达式的完整版本和查询，另外还有Microsoft的FOR XML、OPENXML扩展和Microsoft的模式版本、XML数据精简（XDR）模式。这一章将讲述以上所列出的最后一个主题——XDR模式。

在第1章中，曾提到Microsoft版本的XML模式与W3C规范中的模式差别，这是因为Microsfot获得初始的规范后就立即采用了它，然后相应地修改它而没有与W3C规范并行发展。本章将会讲述这两种规范的不同之处。

本章包括下列内容：

- 对Microsoft的XDR模式的全面讨论。
- Microsoft的BizTalk框架及其功能。
- Microsoft的XDR模式的注解以及用它们把XML映射到关系数据库的工作原理。
- 数据类型以及它们如何在XML和XDR模式之间进行映射。

5.1　概述

在关系数据库管理系统中，通常创建一个数据库的视图，再用SQL来查询该视图。XDR模式也具有相同的作用：创建关系数据库的XML视图。这个视图可以用XPath表达式进行查询。

如第1章所述，XML模式描述XML文档结构，并且能够对文档中的数据进行限制。SQL Server 2000使用XML数据精简语言创建模式。该语言与模式语言的W3C规范相似，该规范可在http://www.w3.org/TR/2000/CR-xmlschema-0-20001024/处获得。虽然这两种方法实现的目标基本上是相同的，但其语言本身却相差甚远。现在，让我们总览一下XDR模式和XDR语言。

5.2　XDR模式

XDR语言是XML数据（XML-Data）规范所描述的概念的子集。Microsoft的XML Parser（MSXML）的实现使用了基于1998年1月W3C发表的XML-Data Note的XML数据精简语言规范。可参阅http://www.w3.org/TR/1998/NOTE-XML-data-0105/。解析器实现也基于XML文档内容描述（Document Content Description，即DTD），这可以从http://www.w3.org/TR/NOTE-dcd处得到。Microsoft的Internet Explorer 5.0以及后续的版本中的XML模式对与这个DCD描述的功能一致的XML数据子集提供支持，即使在XML语法方面有些许不同。

需要全本请访问 www.ertongbook.com

China-pub.com
下载

5.2.1　元素和属性

正如在W3C模式规范中所描述的，若指定<ElementType…>和<AttributeType…>则分别定义了XDR模式中的元素和属性，提供了元素和属性的定义和类型。元素和属性的实例则在<element…>或<attribute…>标记中声明。

代码清单5-1给出了XDR模式。

代码清单5-1　实例XDR模式

```
<?xml version="1.0"?>
<Schema xmlns="schemas-microsoft-com:xml-data">
  <ElementType name="title" />
  <ElementType name="author" />
  <ElementType name="pages" />
  <ElementType name="book" model="closed">
    <element type="title" />
    <element type="author" />
    <element type="pages" />

    <AttributeType name="copyright" />
    <attribute type="copyright" />
  </ElementType>
</Schema>
```

该模式使用<ElementType>定义了四个元素——<title>、<author>、<pages>和<book>。<book>元素指定单独的元素和组成每个元素的属性。换句话说，它描述该元素的内容模型。内容模型所涉及的方面不仅限于上例中所显示的那些，下一节将讨论它的附加组件。

本例表明每一个book元素包括title、author和pages三个子元素。内容模型由element元素和type属性指定，其中type属性引用了前面定义的element类型。

为了使多个元素共享公共属性的定义，可以使用全局属性，如代码清单5-2所示。该模式为book元素声明了一个属性——copyright。这是通过使用定义属性类型的<AttributeType>元素，然后通过attribute元素声明它来实现的。你可以通过将<AttributeType>元素放置到任何一个<ElementType>上下文的外部来全局地指定<AttributeType>元素。

代码清单5-2　声明全局属性的实例

```
<Schema xmlns:s="urn:schemas-microsoft-com:xml-data">
  <ElementType name="title" content="textOnly"/>
  <ElementType name="authors" content="textOnly"/>
  <AttributeType name="pages" content="textOnly"/>
  <ElementType name="book" order="seq" content="eltOnly">

    <attribute type="pages" />
    <element type="title" />
    <element type="authors" />
  </ElementType>
</Schema>
```

China-pub.com 下载

5.2.2 内容模型

内容模型描述了XML文档中的元素和属性的结构。可以通过使用诸如model、minOccurs、maxOccurs、order、content等属性来做到这一点。我们首先介绍元素的内容模型，然后介绍属性的内容模型。

有关该内容模型讨论的其他部分，我想你应该记住：有关元素、属性和属性值的选择是由公司的业务规则严格和完全控制的，没有任何选择。

回顾代码清单5-2，元素title、author和pages的内容模型是非常容易理解的。因为content属性将元素指定为textOnly，所以这些元素仅包含文本，不包含任何子元素。

Book元素的内容模型稍复杂一些。它的content属性是eltOnly，这表明book元素可以只包含指定的元素（title、author和pages）。此外，对book元素的每一个实例，子元素顺序必须与在模式中指定的顺序相同。代码清单5-3是符合代码清单5-2模式的一个实例文档。

代码清单5-3 符合代码清单5-2模式的一个有效的实例文档

```
<msx:book xmlns:msx="x-schema:BookSchema.xml">
  <msx:pages>522</msx:pages>
  <msx:title>Java and XML: A Marriage That Works</msx:title>
  <msx:authors>John Griffin</msx:authors>
<msx:/book>
```

现在看一下内容模型使用的不同属性。

1. model属性（开放和封闭内容模型）

元素的内容模型可以具有开放或封闭的属性。在开放式内容模型中，元素可以有文档所参考的模式中未声明的附加的子元素和属性。与之相反，在封闭模型中，文档不包含不遵循参考模型规定的任何数据。

XDR模式内容模型的缺省值是开放的。这样可以保证XDR模式在文档类型定义（Document Type Declaration，即DTD）中未表现出的可扩展性。DTD是一个封闭模型，它不允许包含任何不遵循规则的信息。

代码清单5-4是前面代码清单5-2中的XML文档的片段。

代码清单5-4 BookSchema片段

```
<msx:book xmlns:msx="x-schema:BookSchema.xml"
        xmlns:msy="urn:some-new-namespace">
  <msx:title msy:id="123"> Java and XML: A Marriage That Works
</msx:title>
  <msx:authors> John Griffin </msx:authors>
  <msx:pages>474</msx:pages>
  <msy:publisher>New Rider's Publishing</msy:publisher>
</msx:book>
```

缺省情况下，这种模式指定一个开放的内容模型。即使还有模式中没有指定的其他元素和属性（比如“urn:some-new-namespace”名字空间（namespace）中定义的title元素的id属性、

China-pub.com
下载

publisher子元素等），该片段也是有效的。

虽然开放内容模型有一定的自由度，可以实现 DTD中不能实现的某些功能，但它存在一些必须遵守的限制：

- 内容不可以被插入或删除，因为这样会破坏已存在的内容模型。例如，模式将 book元素定义为三个元素的一个序列。因此，在添加任何开放内容之前必须提供准确的元素顺序。所以，不能移去pages元素，也不能有两个相邻的title元素。
- 可以添加未声明的元素（只要它们在不同的名字空间中定义）。
- 在提供了模式内容模型之后就可以添加其他元素。例如，即使在 page元素之后又添加一个title元素，一个XML文档仍然会有效。

下面的例子指定一个封闭的内容模型：

```
<x:ElementType name="book" model="closed">
```

正如模式所指示的那样，book元素只包含title、author和pages元素。这样，在前面的XML片段中使用扩展的元素会使文档无效。

2. content属性

content属性描述了元素可以包含的内容。表 5-1列出了可能的值。

表5-1　content属性

值	描　述
textOnly	元素只能包含文本
eltOnly	元素只能包含其他元素
mixed	元素可以包含混合的文本和其他元素
empty	元素必须为空

回顾一下代码清单 5-4所举的例子。正如我们所说过的那样， XDR模式定义了一个包括三个子元素：title、author和pages的book元素。这些子元素只能包含文本，因为这些元素的 content属性都是textOnly，然而，book元素只能包含元素而不包含文本，因为其 content属性为eltOnly。

代码清单5-5再次给出了整个模式。

代码清单5-5　Book XDR模式

```
<?xml version="1.0"?>
<Schema xmlns:s="schemas-microsoft-com:xml-data">
  <s:ElementType name="title" content="textOnly" />
  <s:ElementType name="author" content="textOnly" />
  <s:AttributeType name="pages" content="textOnly" />
  <s:ElementType name="book"  content="eltOnly" model="closed">
    <s:element type="title" />
    <s:element type="author" />
    <s:attribute type="pages" />

    <s:AttributeType name="copyright" />
    <s:attribute type="copyright" />
  </s:ElementType>
</Schema>
```

China-pub.com 下载

还注意到book元素使用封闭的内容模型（model="closed"），这表明book元素只能包含三个子元素，而不能再包含文本或另外的子元素和属性。

empty内容属性规定元素不能包含任何文本或子元素。不同的是，它可以包含属性。而内容属性为mixed的元素可以包含文本和子元素。

3. minOccurs和maxOccurs属性

minOccurs和maxOccurs是指定子元素可以在它的父元素中出现次数的约束规则。这些属性的完整描述见表5-2，可以这样指定它：

```
<element type="Item" maxOccurs="*" />
```

表5-2 约束规则

属 性	描 述	有效值	解 释	缺 省 值
maxOccurs	指定子元素可以在父元素内出现的最大次数	“1”	允许且仅允许该子元素的一个实例	当content=mixed时，缺省值是“*”，其余情况缺省值是“1”
		“*”	该子元素可以出现任意次	n/a
minOccurs	指定子元素可以在父元素内出现的最小次数	“0”“1”	实际上，将该属性指定为零，可以使子元素随意出现	缺省值是“1”
				n/a

代码清单5-6的模式中，author元素将maxOccurs设置为”*”，而pages元素将minOccurs设置为“0”。

代码清单5-6 带有maxOccurs和minOccurs的模式

```
<?xml version="1.0"?>
<Schema xmlns:s="schemas-microsoft-com:xml-data">
  <s:ElementType name="title" content="textOnly" />
  <s:ElementType name="author" content="textOnly" />
  <s:ElementType name="pages" content="textOnly" />
  <s:ElementType name="book"  content="eltOnly" model="closed">
    <s:element type="title" />
    <s:element type="author" maxOccurs="*" />
    <s:element type="pages" minOccurs="0" />
  </s:ElementType>
  <s:ElementType name="root" >
    <s:element type="book" />
  </s:ElementType>
</Schema>
```

代码清单5-6中的模式使得代码清单5-7的文档成为一个有效的XML文档。

代码清单5-7 基于代码清单5-6模式的有效文档

```
<root>
  <book>
    <title>C Programming</title>
    <author>Author A</author>
    <author>Author B</author>
```

需要全本请访问 www.ertongbook.com

China-pub.com
下载


```
    <pages>300</pages>
  </book>
  <book>
    <title>Java Programming</title>
    <author>Author C</author>
  </book>
</root>
```

4. minLength和maxLength属性

名字空间”urn:schemas-microsoft-com:datatypes”指定元素或属性的数据类型。该名字空间定义的minLength和maxLength属性可以限制string、number、bin.hex或bin.base64数据类型的长度。在解析时和运行时这些属性被强制使用，它们有有效的父元素：<ElementType/>、<AttributeType/>和<dataType/>。

在字符串和数字数据类型中，maxLength规定最大字符数，minLength规定最小字符数。

对于bin.hex和bin.base64，maxLength设置二进制对象的最大字节数，minLength设置最小字节数。

下面举例说明如何使用这些长度属性。下面的模式（见代码清单5-8）为userID、password和LoginInfo元素指定内容模型。Password元素指定最小长度是6个字符，最大长度是8个字符。本章的5.4.12节会介绍dt:type及其应用。现在，我们将说明它如何指定一个元素的数据类型。

代码清单5-8　使用minLength和maxLength的实例模式

```
<?xml version="1.0"?>
<Schema xmlns:s="schemas-microsoft-com:xml-data"
        xmlns:dt="urn:schemas-microsoft-com:datatypes" >
  <s:AttributeType name="userID"
        xmlns:dt="urn:schemas-microsoft-com:datatypes"
        dt:type="string" />
<s:AttributeType name="password"
          dt:type="string"
          dt:minLength="6"
          dt:maxLength="8"/>
<s:ElementType name="LoginInfo" >
  <s:attribute type="userID" />
  <s:attribute type="password" />
</s:ElementType>
</Schema>
```

下面是前述模式有效实例文档的两个例子：

```
<LoginInfo userID="1" password="xyz123"/>
<LoginInfo userID="2" password=""/>
```

虽然第二个实例中的password的属性值比minLength短，但该实例仍然有效，这是因为将password属性指定为“ ”和没有规定password属性是一样的。

5. order属性

order属性指定元素在文档实例中出现的顺序。它可以取以下值：“seq”、“one”或“many”。

China-pub.com 下载

若order属性值为“ seq ”，那么所包括的元素在实例文档中出现的顺序必须与模式中的顺序相同。参见代码清单5-9中的模式片段。

代码清单5-9 order=“ seq ”的一个例子

```
<ElementType name="PurchaseOrder" order="seq">
  <element type="PONumber" />
  <element type="PODate" />
  <element type="ShipAddress" />
</ElementType>
```

order属性值中的“ one ”是指“ 或 ”组合。即，只有在一个 <ElementType>中定义的这些子元素之一才能出现在一个实例文档中。所以若将 Item元素指定为包含一个 product元素或backOrderedProduct元素，但不能两者都包括，就可以用下列模式片段来指定这种模式。

```
<ElementType name="Item" order="one">
  <element type="product" />
  <element type="backOrderedProduct" />
</ElementType>
```

order的“ many ”属性值是指子元素可以任何顺序和任何数量出现。

order属性的缺省值是由使用中的内容模型决定的。若内容属性是“ eltOnly ”，order的缺省值就是“ seq ”。若内容属性是“ mixed ”，order的缺省值就是“ many ”。

order属性可以在<ElementType>或group元素中使用。

6. group元素

group元素使我们能够对子元素的一个子集指定限制。这个功能非常有用。

元素group有order、minOccurs、maxOccurs属性。

让我们看看代码清单5-10中的另一个例子。下面的模式将Item元素定义为包含有两个子元素product和backOrderedProduct的group元素。因为group元素的order属性值是“ one ”，所以只能有一个子元素出现在Item元素中。这样，就不可能出现使用 product和backOrderedProduct两个元素的Item元素。这两个子元素中只能出现一个。

代码清单5-10 <group>元素的一个例子

```
<ElementType name="Item">
    <group order="one">
        <element type="product" />
        <element type="backOrderedProduct" />
    </group>
    <element type="quantity"/>
    <element type="price"/>
</ElementType>
```

该模式下使代码清单5-11中的文档有效。

代码清单5-11 符合在代码清单5-10中的模式的文档

```
<Item>
```

China-pub.com
下载


```
    <product>CD</product>
    <quantity>100</quantity>
    <price>10</price>
  </Item>
  <Item>
    < backOrderedProduct >FloppyDisk</ backOrderedProduct >
    <quantity>100</quantity>
    <price>1</price>
  </Item>
```

7. 属性的内容模型

元素<AttributeType>指定元素内部使用的属性类型。必需的属性可以用 required关键字来指定，举例如下：

```
<AttributeType name="shipTo" dt:type="idref" required="yes"/>
```

元素<attribute>指定在元素<AttributeType>中定义的一个属性的实例。它在 <ElementType>元素中使用。

在某些方面，属性比元素受到限制更多。

- 属性不能包含子元素。
- 属性没有“ 或 ”选择。
- 不能指定属性顺序。
- 一个属性在一个元素中只能出现一次（即使将它指定为 required）。

同时，属性也能做一些元素不能做到的事情：

- 属性可以将它的合法值限制在一组字符串里，比如：

```
<AttributeType name="priority" dt:type="enumeration" dt:values="high medium low"/>
```

- 属性有缺省值，如：

```
<AttributeType name= "quantity" dt:type="int">
<attribute type="quantity" default="1"/>
```

虽然不同元素类型可以有同名属性，但这些属性是独立的并且是不相干的。

8. 指定属性的缺省值

属性default指定属性的缺省值。它在模式中的 <AttributeType>和<attribute>元素中指定。

例如：代码清单5-12将属性City的缺省值设为“ Seattle ”。

代码清单5-12　为属性指定缺省值

```
<?xml version="1.0" ?>
<Schema xmlns="urn:schemas-microsoft-com:xml-data" >
<ElementType name="Customer" >
    <AttributeType name="CustomerID" />
    <AttributeType name="ContactName" />
    <AttributeType name="City" default="Seattle" />

    <attribute type="CustomerID" />
    <attribute type="ContactName" />
    <attribute type="City"  />
```

China-pub.com 下载

```
</ElementType>
</Schema>
```

以附录A的Northwind数据库为例，在一个<Customer>元素没有City属性的文档实例中，使用缺省值“Seattle”，文档仍然有效。例如：

```
<Customer CustomerID="ALFKI" ContactName="Maria Anders" City="London"/>
<Customer CustomerID="ANATR" ContactName="Ana Trujillo"/>
```

客户ALFKI显式说明了City属性（“London”），所以忽略缺省值。而客户ANATR没有说明City属性，所以提供了缺省值“Seattle”。

如果模式像下面的例子那样，既提供default属性又提供required属性，那么结果稍有不同。在下面的片段中，<AttributeType>将属性City指定为required，其缺省值为“seattle”。

```
<AttributeType name="City" default="Seattle" required="yes" />
```

元素Customer元素现在必须有一个City属性，而且City属性的值必须为“Seattle”。

5.2.3 数据类型

数据类型规范是模式的必需部分。实际上，数据类型规范是数据类型创建的主要驱动力。W3C XML 1.0 Recommendation（W3C XML 1.0推荐）中定义了枚举（enumeration）类型和一组标记（tokenized）类型。Microsoft的XML文档将这些类型作为主要类型。

表5-3列出了W3C XML 1.0推荐中的3.3.1节定义的基本数据类型。

表5-3 Microsoft的主要数据类型

基本数据类型	描　述
entity	表示XML ENTITY类型
entities	表示XML ENTITIES类型
enumeration	表示枚举类型（只对属性支持）
id	表示XML ID类型
idref	表示XML IDREF类型
idrefs	表示XML IDREFS类型
nmtoken	表示XML NMTOKEN类型
nmtokens	表示XML NMTOKENS类型
notation	表示NOTATION类型
string	表示一个string类型

除基本数据类型之外，Microsoft模式规范还列举了许多其他的类型。表5-4列出了这些类型。在本章5.4.12节中，将会详细讲述怎样将这些数据类型映射到SQL Server数据类型上以及使用哪些SQL服务器的转换函数来翻译XML数据类型。

表5-4 Microsoft支持的XML非基本数据类型

数据类型	描　述
bin.base64	MIME类型、Base64编码的二进制大对象（binary large object，即BLOB）

China-pub.com
下载

（续）

数据类型	描 述
bin.hex	由十六进制数字表示八位字节
boolean	0或1，0=“false”，1=“true”
char	字符串，一个字符的长度
date	ISO 8601格式的子集中的数据，除去时间数据那部分（如“1994-11-05”）。数据本身没有意义（如2-31-99也会通过有效性检查）
dateTime	ISO 8601格式的子集中的数据，时间可选，区域不可选。可以精确到十亿分之一秒（如：“1998-04-07T18:39:09”）
dateTime.tz	ISO 8601格式的子集中的数据，时间和区域都可选。可以精确到十亿分之一秒（如:1988-04-07T18:39:09-08:00）
fixed.14.4	与number相同，但小数点左边不得超过14位，小数点右边不得超过4位
float	位数没有限制的实数。可有前导符、小数位以及指数。标点同美式英语。值的范围是从1.7976931348623157E+308到2.2250738585072014E-308
int	可以有可选符号，但不能有小数位和指数的数字
number	对数字没有限制，可以有前导符、小数位和指数。标点同美式英语（值的范围是从1.7976931348623157E+308到2.2250738585072014E-308）
time	ISO 8601格式的子集中的Time，没有日期和区域（如:“08:15:27”）
time.tz	ISO 8601格式的子集中的Time，没有日期但区域可选（如“08:1527-05:00”）
i1	一个字节表示的整数。符号可选，没有小数和指数部分（如:“1，127，-128”）
i2	一个字表示的整数。符号可选，没有小数和指数部分（如:“1，703，-32768”）
i4	四个字节表示的整数。符号可选，没有小数和指数部分（如:“1，703，-32768，148343，-1000000000”）
i8	八个字节表示的整数。符号可选，没有小数和指数部分，有19位精度。范围是从-9，223，372，036，854，775，808到9，223，372，036，854，775，807
r4	有七位精度的实数；可以有前导符、小数位，指数可选。标点同美式英语。取值范围从3.40282347E+38F到1.17549435E-38F
r8	与float相同。具有15位精度；可有前导符、小数位，指数可选。标点同美式英语。取值范围从1.7976931348623157E+308到2.2250738585072014E-308
ui1	无符号整数。无符号、无小数位、无指数（如:“1，255”）
ui2	两个字节的无符号整数。无符号、无小数位、无指数（如:“1，255，65535”）
ui4	四个字节的无符号整数。无符号、无小数位、无指数（如:“1，703，3000000000”）
ui8	八个字节的无符号整数。无符号、无小数位、无指数，取值范围从0到18，446，744，073，709，551，615
uri	统一资源标识符。（如:“urn:schemas-microsoft-com:office9”）
uuid	表示八位字节的十六进制数。忽略内部的连字符（如:“333C7BC4-460F-11D0-BC04-0080C7055A83”）

在了解了Microsoft版本的XML模式文档之后，接下来，我们在它和SQL Server 2000之间建立连接，并讨论如何使它们协调工作

5.2.4 映射模式

怎样将从数据库得到的相关数据转换成XML文档格式呢？首先从XML模式文档入手。但这还不够。模式描述了文档的结构，但文档和数据之间并没有关系，需要依靠XDR模式中的注解来将两者联系起来。被注解的XDR模式称为映射模式，因为它将模式映射成为关系数据库。这

China-pub.com 下载

样就可以在XML文档的表单中检索数据了（见图5-1）。

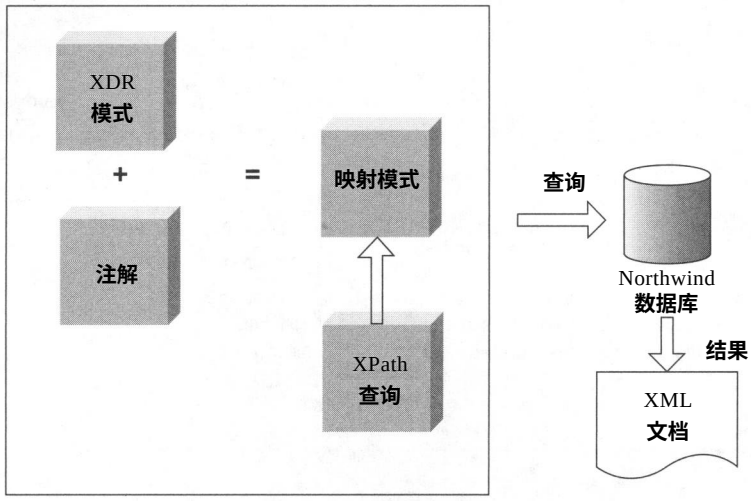


图5-1 模式、注解和XPath

Microsoft定义的几种模式注解将XML数据的元素和属性映射为数据库的表和列。本章的其余小节将分析每一个注解。

将映射模式看作用CREATE VIEW命令创建的数据库视图。可以通过XPath查询来对映射模式进行查询，这在很多地方与用SQL查询从数据库得到数据是相同的。

5.2.5 名字空间

在映射模式中要用到两种不同的名字空间。一个是注解，另一个是数据类型。

注解名字空间在<schema>声明中用“urn:schemas-microsoft-com:xml-sql”来声明。同时可以用sql作为名字空间的前缀，但也可以像上例那样不这样使用。代码清单5-13是一个声明的例子。

代码清单5-13 注解名字空间声明实例

```
<?xml version="1.0" ?>
<Schema xmlns="urn:schemas-microsoft-com:xml-data"
        xmlns:sql="urn:schemas-microsoft-com:xml-sql"
>
...
</Schema>
```

数据类型名字空间也在<schema>声明中用“urn:schemas-microsoft-com:datatypes”声明。要像代码清单5-14那样用dt作为名字空间的前缀。

代码清单5-14 数据类型名字空间声明实例

```
<?xml version="1.0" ?>
<Schema xmlns="urn:schemas-microsoft-com:xml-data"
```

需要全本请访问 www.ertongbook.com

China-pub.com
下载


```
        xmlns:sql="urn:schemas-microsoft-com:xml-sql"
        xmlns:dt="urn:schemas-microsoft-com:datatypes"
>
...
</Schema>
```

5.2.6　带注解的模式例子

本节我们将学以致用。首先看一下代码清单5-15中Northwind数据库的Orders表的XDR模式。

代码清单5-15　Orders表的XDR模式

```
<?xml version="1.0" ?>
<Schema xmlns="urn:schemas-microsoft-com:xml-data"
        xmlns:dt="urn:schemas-microsoft-com:datatypes"
        xmlns:sql="urn:schemas-microsoft-com:xml-sql">

<ElementType name="Orders" >
    <AttributeType name="OrderID" />
    <AttributeType name="CustomerID" />
    <AttributeType name="ShipName" />

    <attribute type="OrderID"/>
    <attribute type="CustomerID" />
    <attribute type="ShipName" />
</ElementType>
</Schema>
```

这里并没有什么新内容。现在像代码清单5-16那样通过加入适当的注解将模式映射到Orders表中。不用担心注解本身的含义，我会解释每一个注解的作用。本章下一节将详细介绍它们。

代码清单5-16　将模式映射为Orders表

```
<?xml version="1.0" ?>
<Schema xmlns="urn:schemas-microsoft-com:xml-data"
        xmlns:dt="urn:schemas-microsoft-com:datatypes"
        xmlns:sql="urn:schemas-microsoft-com:xml-sql">

<ElementType name="Order" sql:relation="Orders">
    <AttributeType name="OrdID" />
    <AttributeType name="CustID" />
    <AttributeType name="SName" />

    <attribute type="OrdID" sql:field="OrderID"/>
    <attribute type="CustID" sql:field="CustomerID"/>
    <attribute type="SName" sql:field="ShipName"/>
</ElementType>
</Schema>
```

本例使用了两个注解：sql:relation和sql:field。sql:relation注解将Order元素映射到Orders表，sql:field注解分别将OrdID、CustID和SName属性映射为OrderID、CustomerID和ShipName域。

China-pub.com 下载

5.2.7 缺省的XDR元素和属性的映射

带注解的XDR模式的缺省行为是将元素映射为同名表或视图。同样，将属性映射为表或视图的同名列。

如果愿意，这些行为是可以修改的。即：可以将元素映射为列。但就像主航线的费用会很高一样，这种映射方法也会有很多的限制。

- 被映射为列的元素不能是复杂类型（不能有子元素）。
- 元素的content属性必须为textOnly。
- 如果没有说明content属性，那么sql:field注解必须显式地将元素指定为列映射（参见本章关于sql:field的讨论）。

现在通过示范在SQL Server上运行这些模式文件的结果来证实前面所讲的内容。首先，将代码清单5-16的模式文件保存为模板目录中的 OrderSchema.xml，该模板目录是在虚目录安装中指定的。然后，创建代码清单 5-17中给出的模板文件，并在相同的模板虚目录中保存为OrderTemplate.xml。如果有必要执行虚名字和目录设置，则参见第 3章。还可以参考第4章中有关模板文件及其执行方法的信息。

代码清单5-17 调用Order模式文件的模板文件

```
<ROOT xmlns:sql="urn:schemas-microsoft-com:xml-sql">
  <sql:Xpath-query mapping-schema="../OrderSchema.xml">
    /Orders
  </sql:xpath-query>
</ROOT>
```

然后使用下面的URL执行OrderTemplate.xml：

http://iisserver/Nwind/templates/OrderTemplate.xml

首先返回的五个记录在代码清单 5-18中列出。

代码清单5-18 执行模板文件的部分结果

```
<ROOT xmlns:sql="urn:schemas-microsoft-com:xml-sql">
  <Orders OrderID="10248" CustomerID="VINET" ShipName="Vins et alcools
   Chevalier" />
  <Orders OrderID="10249" CustomerID="TOMSP" ShipName="Toms
   Spezialitäten" />
  <Orders OrderID="10250" CustomerID="HANAR" ShipName="Hanari Carnes" />
  <Orders OrderID="10251" CustomerID="VICTE" ShipName="Victuailles en
   stock" />
  <Orders OrderID="10252" CustomerID="SUPRD" ShipName="Suprêmes délices"
/>
...
</ROOT>
```

注意两点：首先也是最重要的一点，将模式文件存入模板目录的原因是：模板文件将模式文件的位置指定在同一目录下。当将模式文件放入模式目录时，需要说明模板目录中的模式文

China-pub.com
下载

件的相对地址。这是指将 <sql:xpath-query mapping-schema="OrderSchema.xml">转换为 <sql:xpath-query mapping-schema="../schemas/OrderSchema.xml">。

如果系统管理员允许对服务器进行直接路径访问（我认为他们会这样做），那么也可以指定绝对路径，如：

c:\inetpub\wwwroot\Nwind\schemas\OrderSchema.xml。

全部讨论建立在假设两个目录在目录树上是相互并行的基础上，正如在第3章建立的结构那样。

另外，代码清单5-15的第三行的/Orders是一个返回所有找到的订单的XPath查询。

在详细解释每一个注解之前，我需要用一部分内容来谈一下 Microsoft针对几个目前困扰开发者的比较突出的分布计算问题提出的解决方法。这些解决方法使用 XML和模式作为它的语言。Microsoft称这种模式为BizTalk。

5.3 BizTalk

随着当今每个交叉于应用程序或业务边界的分布对象技术的出现，与提供有效、自动地交互相关联的新的挑战具有举足轻重的作用。

随着当今Internet上电子商务应用的迅速发展，与其他组织联系的安全性和可靠性的要求越来越强。

这些涉及到业务边界、业务之间（interbusiness）或部门的挑战至少包括以下几点：

- 用来通过应用程序或业务边界指定、打包、出版和交换结构化或非结构化信息的灵活、可扩展的公用语言。
- 在跨应用、跨业务边界时，用来将信息从一种格式转换为另一种格式的灵活、可扩展的公用语言。
- 支持跨应用和跨业务边界的平台无关的应用级通信协议。
- 提供信息安全性、完整性和保密性的平台无关机制。

Microsoft的BizTalk规范是解决这些问题的一个尝试。它为应用程序与组织之间的通信所需的基于XML的信息提供了设计和开发的规范。 BizTalk框架并没有阐述 B-to-B（business-to-business）电子商务的所有方面。它只是为大多数 B-to-B的电子交换提供了一套基本机制。Microsoft依赖于其他与BizTalk框架一致的、为特定应用和特定领域开发的规范和标准。现在，它的作用在某种程度上像一个由第三方组织开发的模式仓库。

现在看一下有代表性的BizTalk文档。

BizTalk模式举例

本讨论并不是对BizTalk的详尽而深刻的论述，而是对 BizTalk文档的组成部分的概述。

BizTalk需要简单对象访问协议（Simple Object Access Protocol，SOAP）提供消息。SOAP的W3C格式的更多信息可以参阅 http://www.w3.org/TR/SOAP/。Microsoft的MSDN网站有很好的关于SOAP的白皮书“ Web services and the Simple Object Access Protocol ”，参见http://msdn.microsoft.com/xml/general/soap_webserv.asp。

China-pub.com 下载

另外，BizTalk没有为特定的业务文档预定义内容或结构（模式）。业务文档模式的细节由涉及到的商家定义和协商。可以将BizTalk看作对业务文档的包装。

代码清单5-19是一个简单的BizTalk文档示例。该文档由标准SOAP 1.1信息组成，包括以下几方面：

- 一个只用于应用程序的业务文档（本例中是购书订单），它具有应用程序规定的由消息体传送的XML名字空间。
- 只用于BizTalk的SOAP头标记<endpoints>和<properties>，由标准BizTag名字空间中定义的BizTag构造而成。BizTalk规范中定义了该模式。

代码清单5-19 一个简单的BizTalk文档

```
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance">
  <SOAP-ENV:Header>
    <eps:endpoints SOAP-ENV:mustUnderstand="1"
        xmlns:eps="http://schemas.biztalk.org/btf-2-0/endpoints"
        xmlns:agr="http://www.trading-agreements.org/types/">
      <eps:to>
        <eps:address xsi:type="agr:department">Book Orders</eps:address>
      </eps:to>
      <eps:from>
        <eps:address xsi:type="agr:organization">Book
Lovers</eps:address>
      </eps:from>
    </eps:endpoints>
    <prop:properties SOAP-ENV:mustUnderstand="1"
        xmlns:prop="http://schemas.biztalk.org/btf-2-0/properties">
      <prop:identity>uuid:74b9f5d0-33fb-4a81-b02b-
5b760641c1d6</prop:identity>
      <prop:sentAt>2000-05-14T03:00:00+08:00</prop:sentAt>
      <prop:expiresAt>2000-05-15T04:00:00+08:00</prop:expiresAt>
      <prop:topic>http://electrocommerce.org/purchase_order/</prop:topic>
      </prop:properties>
    </SOAP-ENV:Header>
    <SOAP-ENV:Body>
      <po:PurchaseOrder
  xmlns:po="http://electrocommerce.org/purchase_order/">
        <po:Title>Essential BizTalk</po:Title>
      </po:PurchaseOrder>
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

总的来说，BizTalk文档主体包括几个相关业务文档，文档头包括几个只用于BizTalk的头表项。若SOAP-ENV:mustUnderstand属性的值为1，则该文档的目标点必须理解和正确处理头表项，否则文档的处理会以失败告终。

所有的BizTags在标准Biztag名字空间内使用URI定义，URI从前缀http://schemas .biztalk .org/btf-2-0/派生而来。

路由标记<to>和<from>通常对源和目标寻址的使用业务实体名而不是HTTP URL之类的传输

China-pub.com
下载

地址。地址内容的格式和翻译由xsi:type属性决定。BizTalk文档结构和功能与文档在什么上面进行传输无关。这种传输无关性是进行BizTalk框架设计时主要考虑的地方。

文档的Body元素包括携带的业务文档。总的来说，BizTalk文档可以携带一组相关的业务文档（例如：定购单、发货人姓名和发送订单的地址）。

除了我在这里提到的BizTalk标记外，还有很多其他BizTalk标记，可以通过BizTalk规范来获得详细的列表和对标记的解释。

BizTalk很有希望达到它的设计目标，使分布式应用的开发不再是一个痛苦的过程。

5.4 XDR模式的注解

有几种注解可以使XDR模式语言将XML文档映射为关系数据库。这些注解规定了XML到关系的映射，包括XML模式的元素和属性到数据库中的表/视图和列的映射。记住以下规则可以事半功倍：注解模式将元素名映射为数据库的表（视图）名，属性名映射为列。注解也可以规定XML（模型数据库关系）的层次关系。

表5-5是一个注解的代码清单。本章将会通过举例逐个详细讲述。

表5-5 模式注解

注 解	描 述
sql:relation	将XML项映射为数据库表
sql:field	将XML项映射为数据库列
sql:is-constant	创建不映射为表但出现在查询结果中的XML元素
sql:map-field	允许从结果中除去模式项
<sql:relationship>	指定XML元素的关系，关系由属性key、key-relation、foreign-key和foreign-relation建立
sql:limit-field	允许限制返回的值
sql:limit-value	基于一个有限值
sql:key-fields	使用惟一标识表中行的列的规范
sql:target-namespace	允许将查询结果中的元素和属性从缺省名字空间移到另一个名字空间
sql:id-prefix	创建有效的XML ID、IDREF和IDREFS。用字符串作为ID、IDREF和IDREFS的值前缀
sql:use-cdata	允许将CDATA规定为可以被XML文档的某些元素使用
sql:url-encode	当XML元素或属性映射为SQL Server BLOB列时，允许返回BLOB数据将会用到的索引
sql:overflow-field	标识包含溢出数据的数据库列

5.2.7节中给出了一个简单的例子，XML模式在选择数据的查询中使用XPath表达式。在第2章中，我们论述了XPath查询的简化形式。现在，为简便起见，我们将继续使用这种简化形式，一直到第6章我们详细讨论XPath表达式的时候。

5.4.1 XDR元素和属性到表和列的显式映射

如前所述，只要元素和属性的名称分别与数据库中的表（视图）和列的名称相同，那么显式映射是没有必要的。注解模式中的元素名映射为指定的数据库的表（视图）名，属性名映射为列名。但在实践中，这种现象很少出现。通常需要为报告之类的应用改变列名。

如果名称不同，那么需要两个注解来指定XML文档中的元素或属性到数据库中的表或列的

China-pub.com 下载

映射。这两个注解为：

- sql:relation。
- sql:field。

下面将分析这两种注解。

1. sql:relation

sql:relation注解显式地将XML元素映射为数据库的表或视图，而且该注解可以加入模式的<ElementType>、<element>或<attribute>节点。注解的一般作用是将SQL Server的非法标识映射为XML节点。例如，SQL Server内一个名为PUB INFO的表是有效的，但它作为一个XML元素却是无效的。可以通过使用sql:relation显式地重映射表名：

```
<ElementType name="PI" sql:relation="[PUB INFO]">
```

在<ElementType>中使用sql:relation会使关系作用到<ElementType>中全部的属性和子元素上去。<AttributeType>忽略该注解。

下面是代码清单5-20列举的例子。sql:relation将模式的<Order>元素映射为Orders表。因为注解作用于<ElementType>，所以所有的属性也都映射到Orders表。然后将该文件存入模式的虚目录中。

代码清单5-20 sql:relation注解举例

```
<?xml version="1.0" ?>
<Schema xmlns="urn:schemas-microsoft-com:xml-data"
        xmlns:dt="urn:schemas-microsoft-com:datatypes"
        xmlns:sql="urn:schemas-microsoft-com:xml-sql">

<ElementType name="Order" sql:relation="Orders">
    <AttributeType name="OrderID" />
    <AttributeType name="CustomerID" />
    <AttributeType name="ShipName" />

    <attribute type="OrderID" />
    <attribute type="CustomerID" />
    <attribute type="ShipName" />
</ElementType>
</Schema>
```

现在用代码清单5-21的模板对该模式进行查询。

代码清单5-21 查询模板

```
<ROOT xmlns:sql="urn:schemas-microsoft-com:xml-sql">
  <sql:xpath-query mapping-schema="../schemas/OrderSchema.xml">
    /Order[@CustomerID="VINET"]
  </sql:xpath-query>
</ROOT>
```

用下面的URL运行模板

http://iisserver/Nwind/templates/OrderTemplate.xml

需要全本请访问 www.ertongbook.com