



WHITE PAPER

Query XML Data Directly from SQL Server 2000

By: Travis Vandersypen, President of DilMad Enterprises, Inc.

Abstract

XML is quickly becoming the preferred method of passing information, not only for the internet, but also across applications, and even within the same application. Until now, developers have been forced to create our own routines to convert data stored within a database system into XML. With the release of SQL Server 2000, however, the potential exists to query data directly from SQL Server into XML format.

With the advent of XML, and the increasing requirements for distributed applications in today's marketplace, a larger demand has been placed on the developer to provide messaging and data in an XML format. Providing data in XML format, until now has consisted of querying data in the form of a cursor or ADO RecordSet, and using a conversion routine to convert the data from into XML format. Now with the new features in SQL Server 2000, this task can be accomplished with minimal effort, which allows the developer to concentrate on the more important task of writing the business logic.

SQL Server 2000 provides a new feature that allows the developer to query SQL Server data and receive that data in XML format through the use of a special clause: FOR XML. This clause provides 3 different options by which SQL Server can return data in XML format: AUTO, RAW, and EXPLICIT.

Issuing the SQL Select command with FOR XML AUTO will return the result set in XML format with each record having a node whose tag name is the same as the table name on which the query was performed. Each node will have attributes equal to the fieldnames specified in the query with values equal to the values of the fields within the table. Using the FOR XML RAW clause will return XML in which each record is represented by a node whose tag name is row and whose attributes are the fields from the query. The last option, FOR XML EXPLICIT, uses queries written in a specific format to return the XML in a specific format. However, these options are only available from within the SQL Server Query Analyzer or by accessing SQL server through a URL.



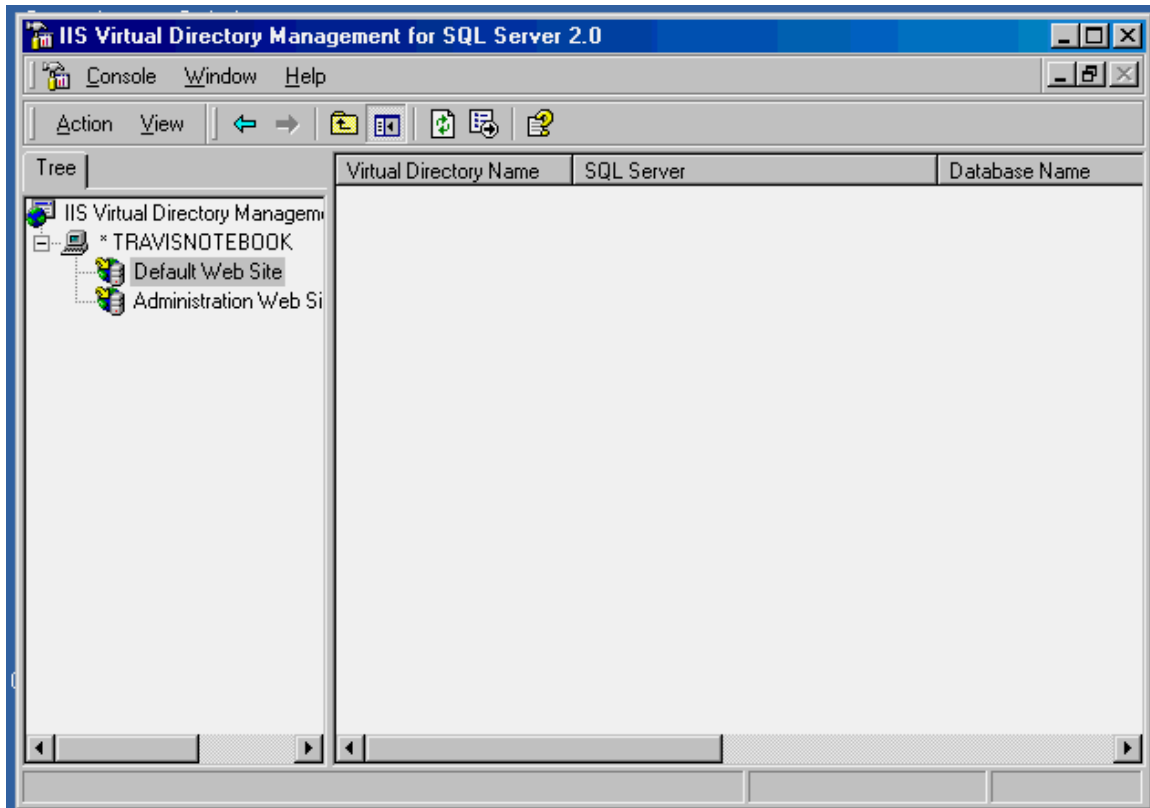
Contents

<i>Configuring SQL Server 2000 for XML Support</i>	<i>3</i>
<i>URL Queries</i>	<i>9</i>
FOR XML AUTO	10
FOR XML RAW	11
Style Sheet Transformations	12
<i>Template Queries</i>	<i>16</i>
Dynamic Template Queries	21
Defining XPath Queries within Template Files	21
<i>XPath Queries</i>	<i>22</i>
Authoring Annotated XSD Schemas	22
XPath Syntax	30
<i>Alternative Ways to Query SQL Server 2000</i>	<i>31</i>
<i>Summary</i>	<i>32</i>

Configuring SQL Server 2000 for XML Support

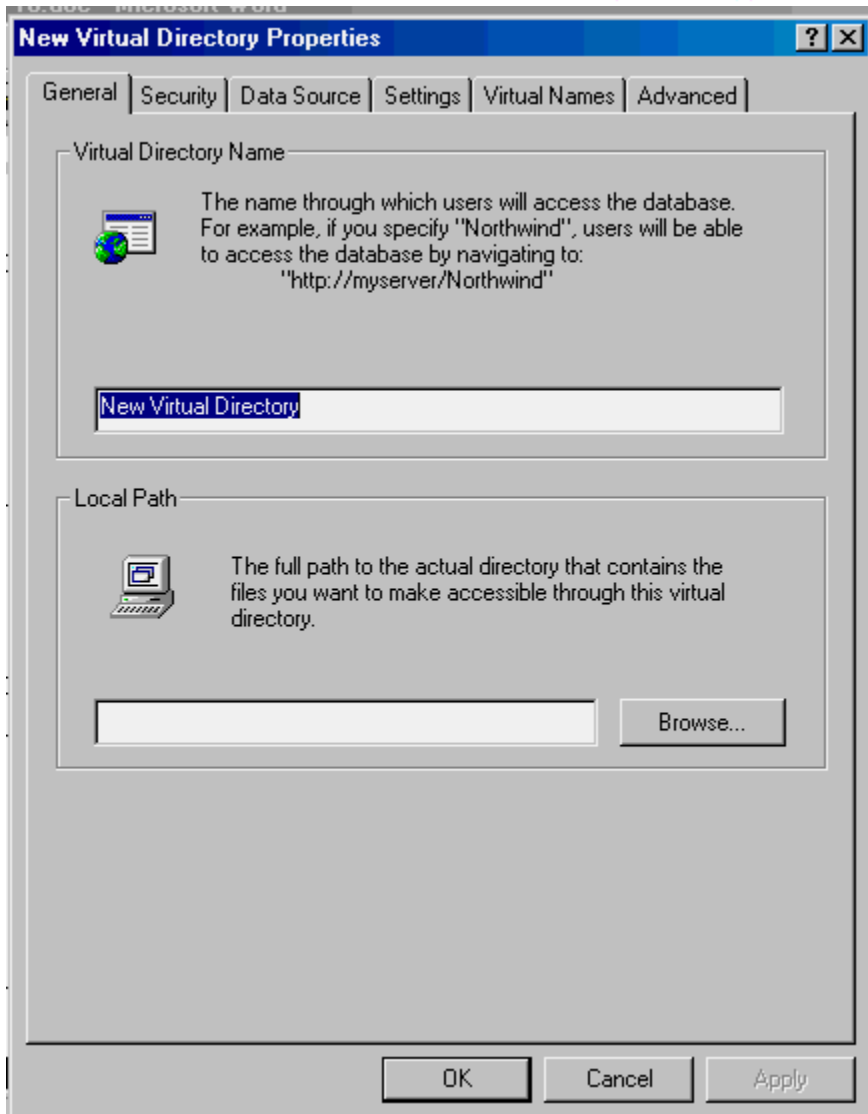
Before you can utilize the new XML support within SQL Server 2000, you must first configure it to do so. Simply click on Start, Programs, Microsoft SQL Server XML Tools, Configure IIS Support – Web Release 2. This will launch the configuration utility necessary to provide XML support within SQL Server 2000 as shown in Figure 1 below.

Figure 1. The IIS Virtual Directory Management Console is where you can administer the XML support for SQL Server.



Once you have the IIS Virtual Directory Management for SQL Server open, drill down in the tree to the left until you get to the Default Web Site node or the web site from which you wish to access SQL Server 2000. On the right side of the screen, right-click and select New, Virtual Directory. This will bring up the New Virtual Directory Properties dialog where you enter information regarding how SQL Server 2000 should be configured to run when accessed from a URL as shown in Figure 2 below.

Figure 2. The General page of the Virtual Directory Properties dialog.



On the first page, you will be required to give the new virtual directory a name and the physical path on the hard drive associated with this virtual directory. This first page determines how you may access SQL Server via HTTP. For instance, in the case of the Default Web Site, if you entered a virtual directory name of SQL2000, you could access it via HTTP as <http://localhost/sql2000>.

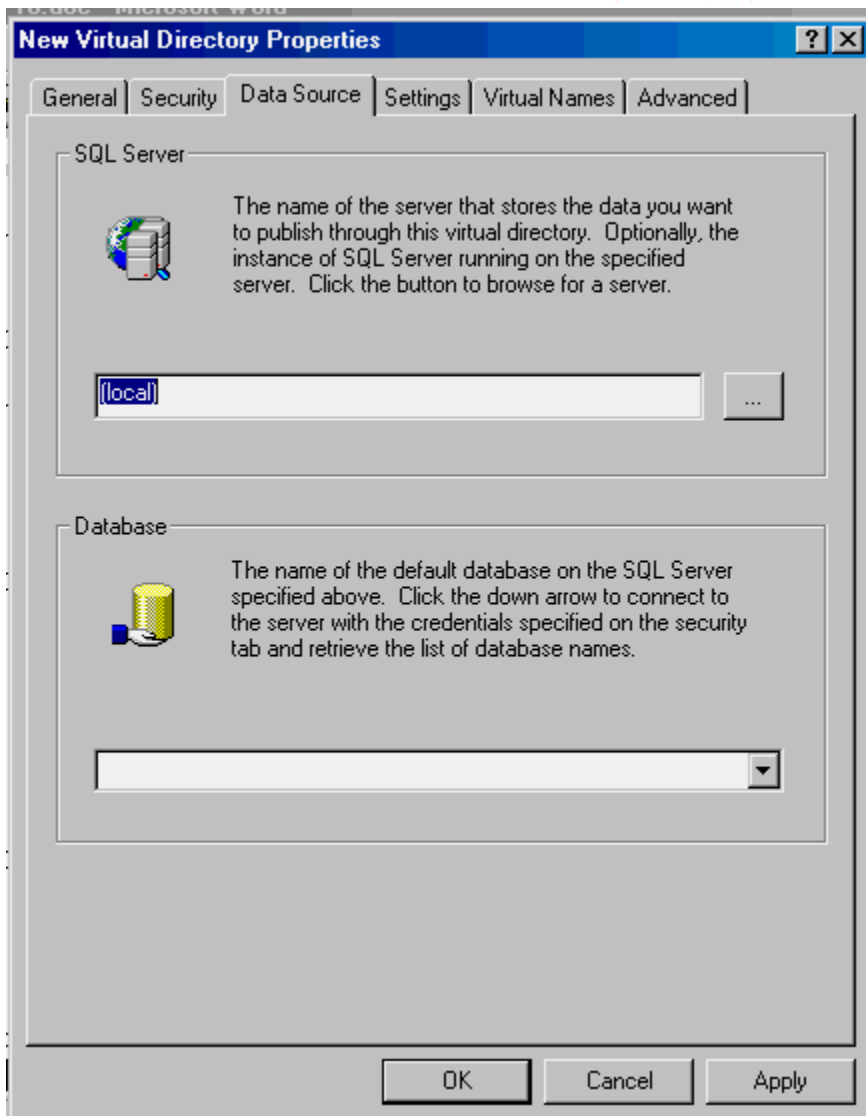
The second page, as shown in Figure 3 below, indicates how you wish to log into SQL Server 2000.

Figure 3. The Security page of the Virtual Directory Properties dialog.



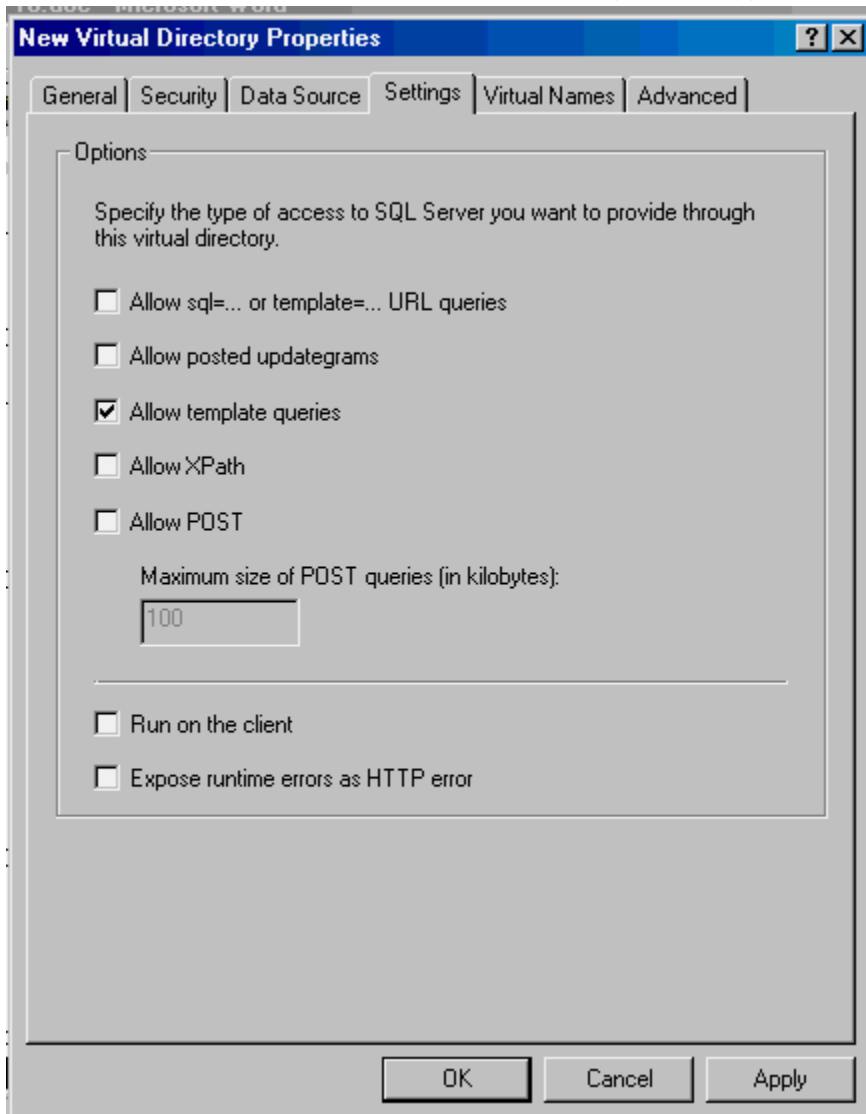
The third page, shown below in Figure 4, allows you to specify the SQL Server installation to use and the database name to access.

Figure 4. The Data Source page of the Virtual Directory Properties dialog.



Page 4 shown below in Figure 5, allows you to indicate the different types of queries that can be run: URL queries, Template queries, and/or XPath queries, as well as whether or not HTTP Posts are allowed.

Figure 5. The Settings page of the Virtual Directory Properties dialog.



The fifth page allows you to map various special virtual directories to your main SQL Server virtual directory. If you want to execute Template Queries, you'll need to at least create a Template virtual directory here, and if you want to use XPath Queries, you'll need to create a Schema virtual directory here also. Figure 6 shows you the fifth page of the dialog, while Figure 7 shows you the Virtual Name Configuration Dialog.

Figure 6. The Virtual Names page of the Virtual Directory Properties dialog.

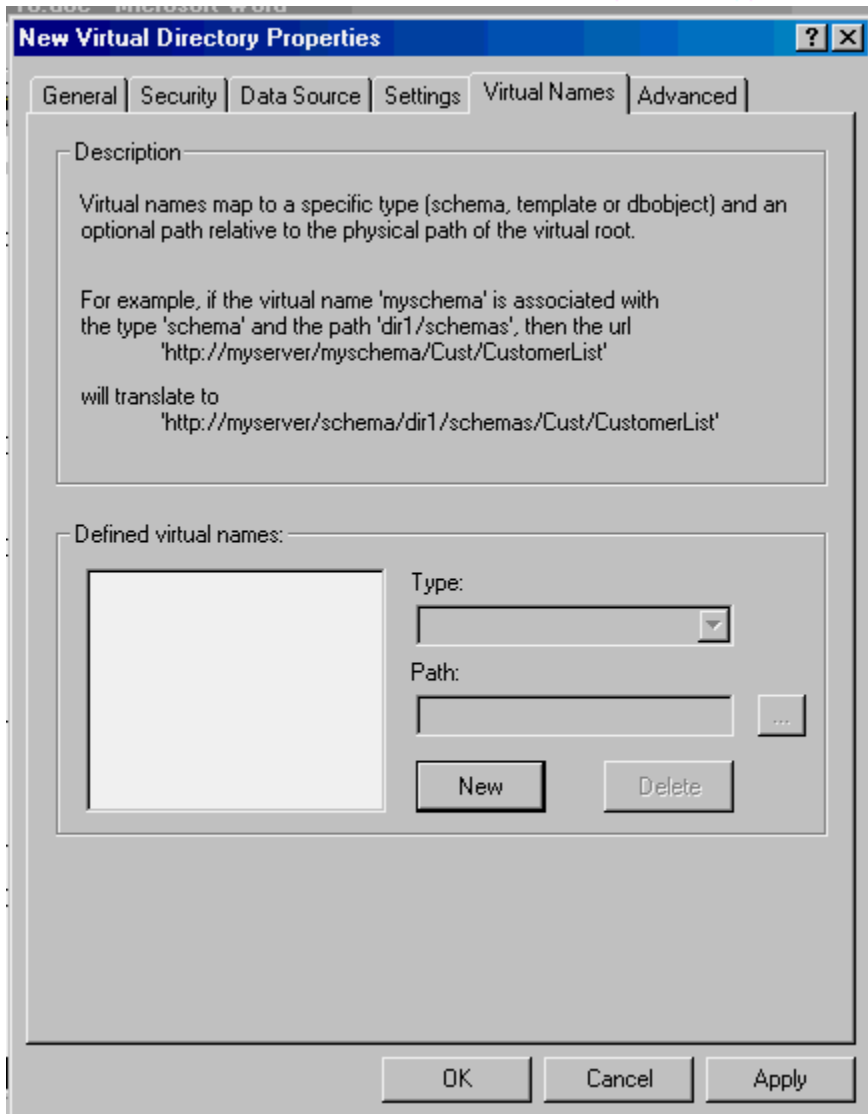
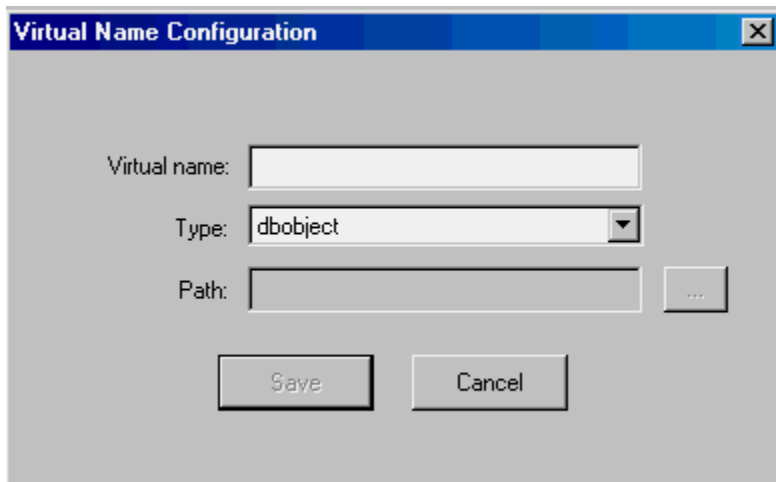
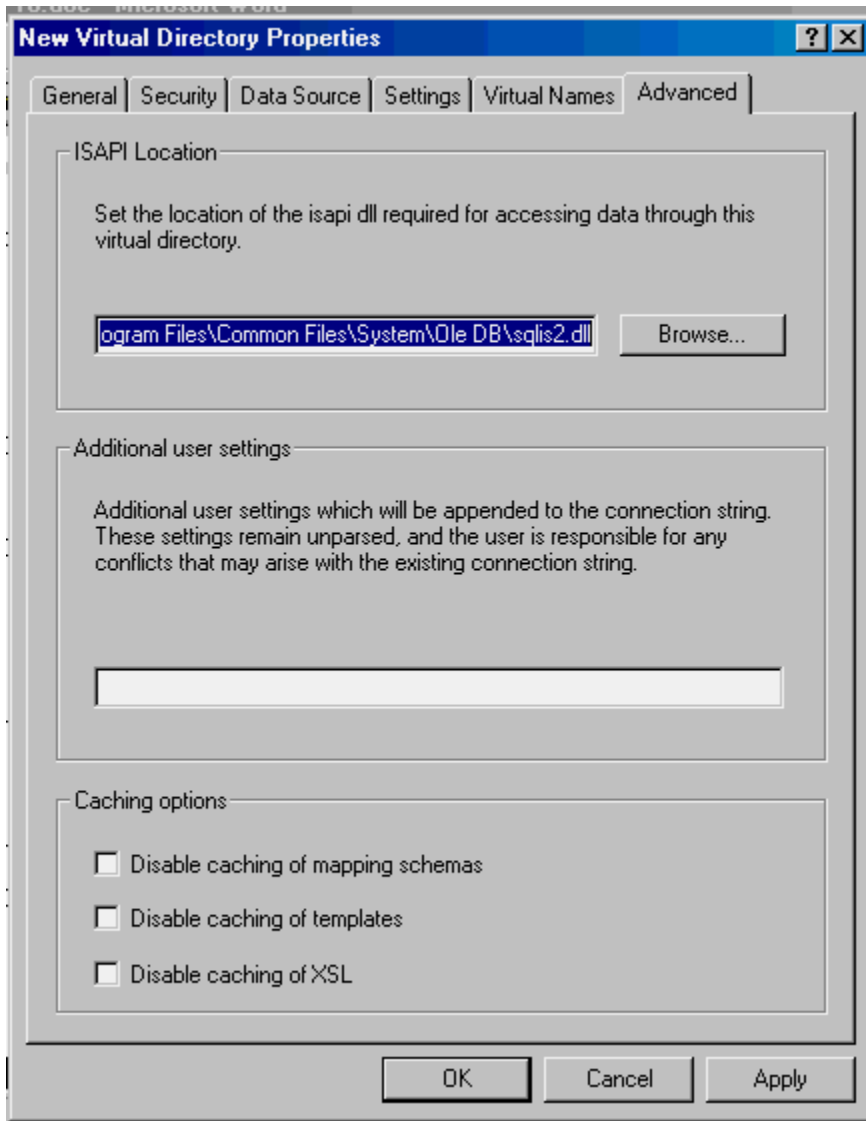


Figure 7. The Virtual Name Configuration dialog.



There is one final page on the dialog for advanced configuration options. This page allows you to specify the SQL ISAPI dll to use for the virtual directory, allows you to specify some additional settings, and whether or not to cache various items in memory, such as Templates, Schemas, and XSLT style sheets. Figure 8, below, shows you what this final page looks like.

Figure 8. The Advanced page of the Virtual Directory Properties dialog.



Once you've configured IIS to support XML for SQL Server 2000, you can begin accessing your new virtual directory via HTTP in the form of URL Queries, Template Queries, and XPath Queries.

URL Queries

The easiest way in which to test or become familiar with URL queries from SQL Server is to open Internet Explorer and enter queries into the Address space available. It is

important to keep in mind however, that the XML string returned by SQL Server 2000 is not "well-formed" XML. This is because there is no single root node from which all other nodes are children. However, a parameter can be passed along with the query itself to specify the root node, which will eliminate this problem by wrapping the returned XML string with the node specified. To return XML data natively from SQL Server 2000, you must include the FOR XML clause in your URL query. This clause can be one of the following:

- AUTO
- RAW
- EXPLICIT

FOR XML AUTO

By specifying **FOR XML AUTO** as part of the URL query, SQL Server 2000 transforms the result set into XML in which each record is represented with a tag for the table name entered in the query and with an attribute on that tag for each field in the result set.

For instance, to the following query:

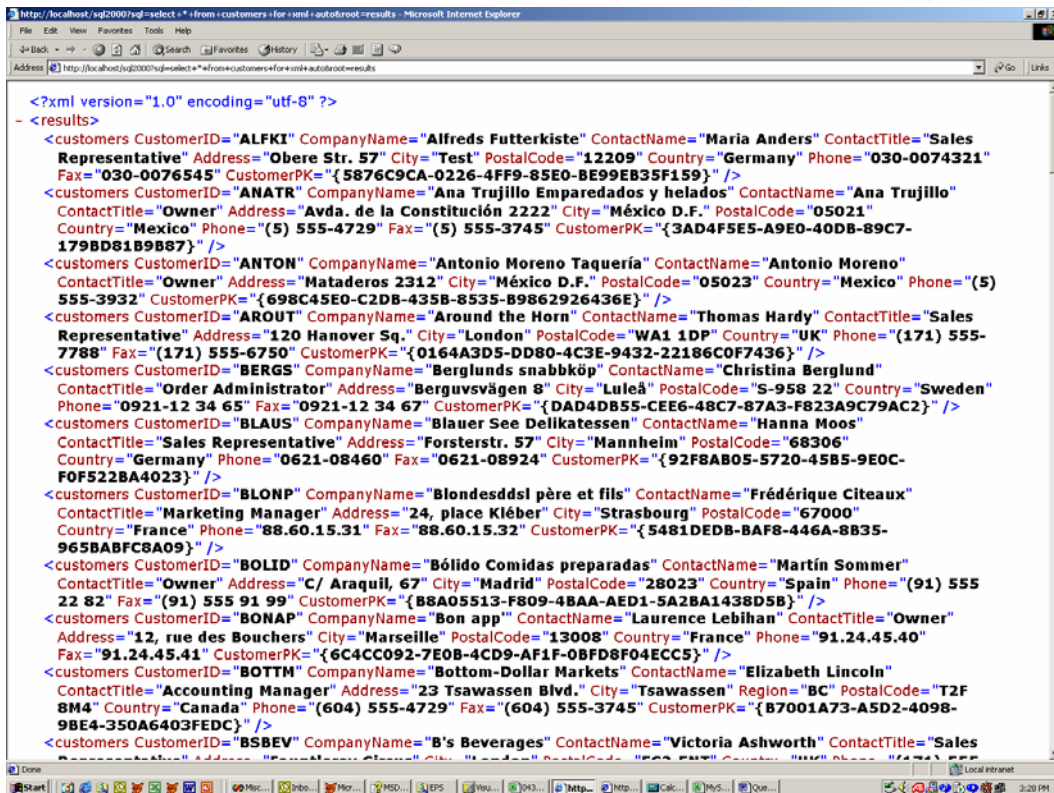
```
SELECT * FROM Customers
```

as a URL query looks like:

```
http://localhost/sql2000?sql=select+*+from+customers+for+xml+auto&root=results
```

which returns all the customers records in the customers table wrapped with a root node called results. Figure 9 is a representative sample of the XML returned from SQL Server 2000.

Figure 9. The results of performing a URL query with For Xml Auto.



```

<?xml version="1.0" encoding="utf-8" ?>
- <results>
  <customers CustomerID="ALFKI" CompanyName="Alfreds Futterkiste" ContactName="Maria Anders" ContactTitle="Sales Representative" Address="Obere Str. 57" City="Test" PostalCode="12209" Country="Germany" Phone="030-0074321" Fax="030-0076545" CustomerPK="{5876C9CA-0226-4FF9-85E0-BE99EB35F159}" />
  <customers CustomerID="ANATR" CompanyName="Ana Trujillo Emparedados y helados" ContactName="Ana Trujillo" ContactTitle="Owner" Address="Avda. de la Constitución 2222" City="México D.F." PostalCode="05021" Country="Mexico" Phone="(5) 555-4729" Fax="(5) 555-3745" CustomerPK="{3AD4F5E5-A9E0-40DB-89C7-179BD81B9887}" />
  <customers CustomerID="ANTON" CompanyName="Antonio Moreno Taquería" ContactName="Antonio Moreno" ContactTitle="Owner" Address="Mataderos 2312" City="México D.F." PostalCode="05023" Country="Mexico" Phone="(5) 555-3932" CustomerPK="{698C45E0-C2DB-435B-8535-B9862926436E}" />
  <customers CustomerID="AROUT" CompanyName="Around the Horn" ContactName="Thomas Hardy" ContactTitle="Sales Representative" Address="120 Hanover Sq." City="London" PostalCode="WA1 1DP" Country="UK" Phone="(171) 555-7788" Fax="(171) 555-6750" CustomerPK="{0164A3D5-DD80-4C3E-9432-22186C0F7436}" />
  <customers CustomerID="BERGS" CompanyName="Berglunds snabbköp" ContactName="Christina Berglund" ContactTitle="Order Administrator" Address="Berguvsvägen 8" City="Luleå" PostalCode="S-958 22" Country="Sweden" Phone="0921-12 34 65" Fax="0921-12 34 67" CustomerPK="{DAD4DB55-CEE6-48C7-87A3-F823A9C79AC2}" />
  <customers CustomerID="BLAUS" CompanyName="Blauer See Delikatessen" ContactName="Hanna Moos" ContactTitle="Sales Representative" Address="Forsterstr. 57" City="Mannheim" PostalCode="68306" Country="Germany" Phone="0621-08460" Fax="0621-08924" CustomerPK="{92F8AB05-5720-45B5-9E0C-F0F522BA4023}" />
  <customers CustomerID="BLONP" CompanyName="Blondesddsl père et fils" ContactName="Frédérique Citeaux" ContactTitle="Marketing Manager" Address="24, place Kléber" City="Strasbourg" PostalCode="67000" Country="France" Phone="88.60.15.31" Fax="88.60.15.32" CustomerPK="{5481DEDB-BAF8-446A-8B35-965BABFC8A09}" />
  <customers CustomerID="BOLID" CompanyName="Bólid Comidas preparadas" ContactName="Martín Sommer" ContactTitle="Owner" Address="C/ Araquil, 67" City="Madrid" PostalCode="28023" Country="Spain" Phone="(91) 555 22 82" Fax="(91) 555 91 99" CustomerPK="{88A0513-F809-4BAA-AED1-5A2BA1438D5B}" />
  <customers CustomerID="BONAP" CompanyName="Bon app" ContactName="Laurence Lebihan" ContactTitle="Owner" Address="12, rue des Bouchers" City="Marseille" PostalCode="13008" Country="France" Phone="91.24.45.40" Fax="91.24.45.41" CustomerPK="{6C4CC092-7E0B-4CD9-AF1F-0BFD8F04ECC5}" />
  <customers CustomerID="BOTTM" CompanyName="Bottom-Dollar Markets" ContactName="Elizabeth Lincoln" ContactTitle="Accounting Manager" Address="23 Tsawassen Blvd." City="Tsawassen" Region="BC" PostalCode="T2F 8M4" Country="Canada" Phone="(604) 555-4729" Fax="(604) 555-3745" CustomerPK="{B7001A73-A5D2-4098-9BE4-350A6403FEDC}" />
  <customers CustomerID="BSBEV" CompanyName="B's Beverages" ContactName="Victoria Ashworth" ContactTitle="Sales Representative" Address="Foothills Blvd." City="Edmonton" PostalCode="T6C 5B7" Country="UK" Phone="(403) 555
  
```

As shown, each record is represented by a node called **customers** whose attributes are the fields within the customers table. One thing to keep in mind is that XML is case-sensitive and as such the XML tag for each record returned by SQL Server will be spelled exactly the way in which it was specified in the query. That means:

```
http://localhost/sql2000?sql=select+*+from+customers+for+xml+auto&root=results
```

returns a different XML grammar than:

```
http://localhost/sql2000?sql=select+*+from+Customers+for+xml+auto&root=results
```

which means that an XSL Pattern match for one will not work for the other.

FOR XML RAW

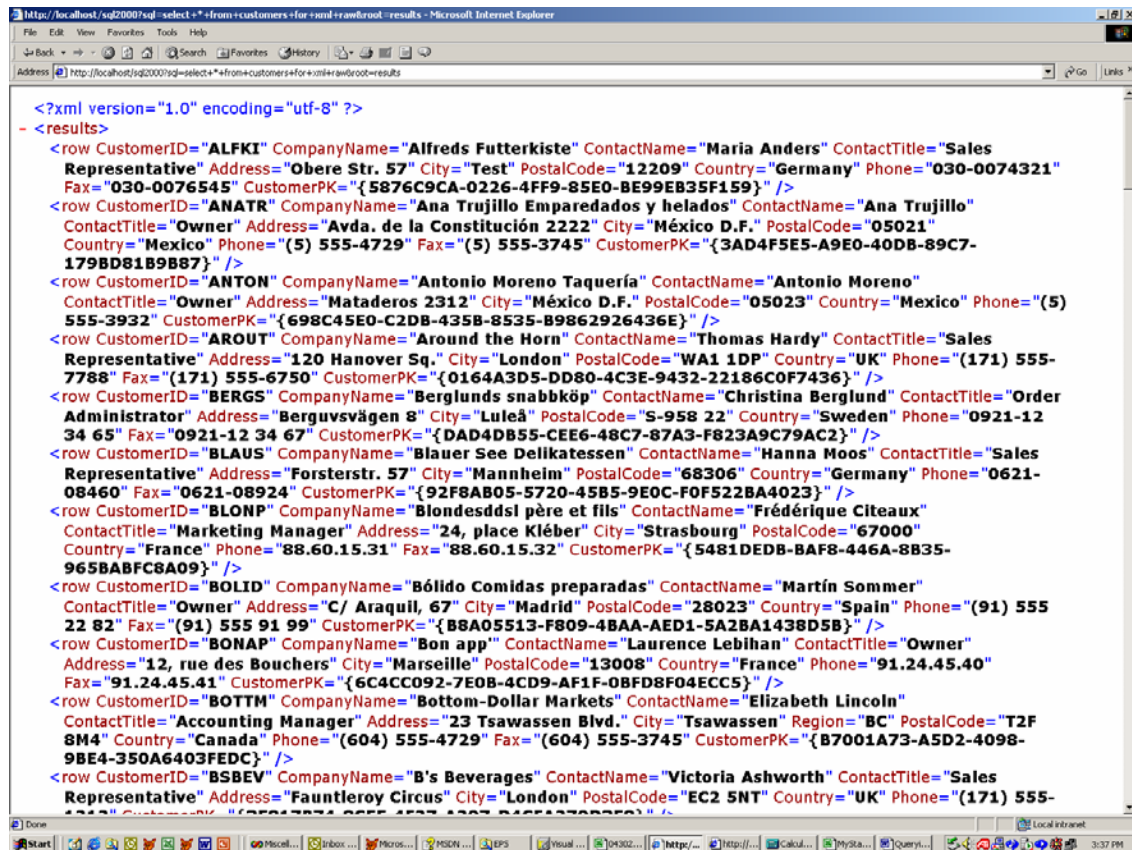
Since that can be a problem at times, SQL Server 2000 allows the definition of a constant row tag named **row**. To make a query against SQL Server 2000 and have it return an XML grammar with row as the tag for each record returned in the result set, simply include the **FOR XML RAW** clause in the URL query.

For instance, navigating to:

```
http://localhost/sql2000?sql=select+*+from+customers+for+xml+raw&root=results
```

returns all records in the customers table with a tag named **results** as the root node. The main difference between this query and the previous ones is that every record is now represented by a tag named **row** as shown in Figure 10, below.

Figure 10. The results of a URL query using For Xml Raw.



Style Sheet Transformations

Both of versions of the returned XML appear very similar to the way in which ADO stores and loads RecordSets in XML format. However, this may not always coincide with the grammar of XML that is expected within an application. For this reason, another parameter, **xsl**, can be included with the URL to specify an XSL style sheet to use to transform the native XML grammar given by SQL Server 2000 into the expected grammar. Let's look at the following example of an XSLT style sheet shown in Listing 1, which, in this case, resides in the root of the Virtual Directory SQL2000:

Listing 1. Customers1.xsl contains a style sheet transformation example.

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
  <xsl:template match="/">
    <CUSTOMERS>
      <xsl:for-each select="Results/Customers">
```

```
<CUSTOMER>
  <CUSTOMERID><xsl:value-of
select="@CustomerID"/></CUSTOMERID>
  <COMPANY><xsl:value-of
select="@CompanyName"/></COMPANY>
  <CONTACT><xsl:value-of
select="@ContactName"/></CONTACT>
  <ADDRESS><xsl:value-of select="@Address"/></ADDRESS>
  <CITY><xsl:value-of select="@City"/></CITY>
  <PHONE><xsl:value-of select="@Phone"/></PHONE>
</CUSTOMER>

</xsl:for-each>
</CUSTOMERS>

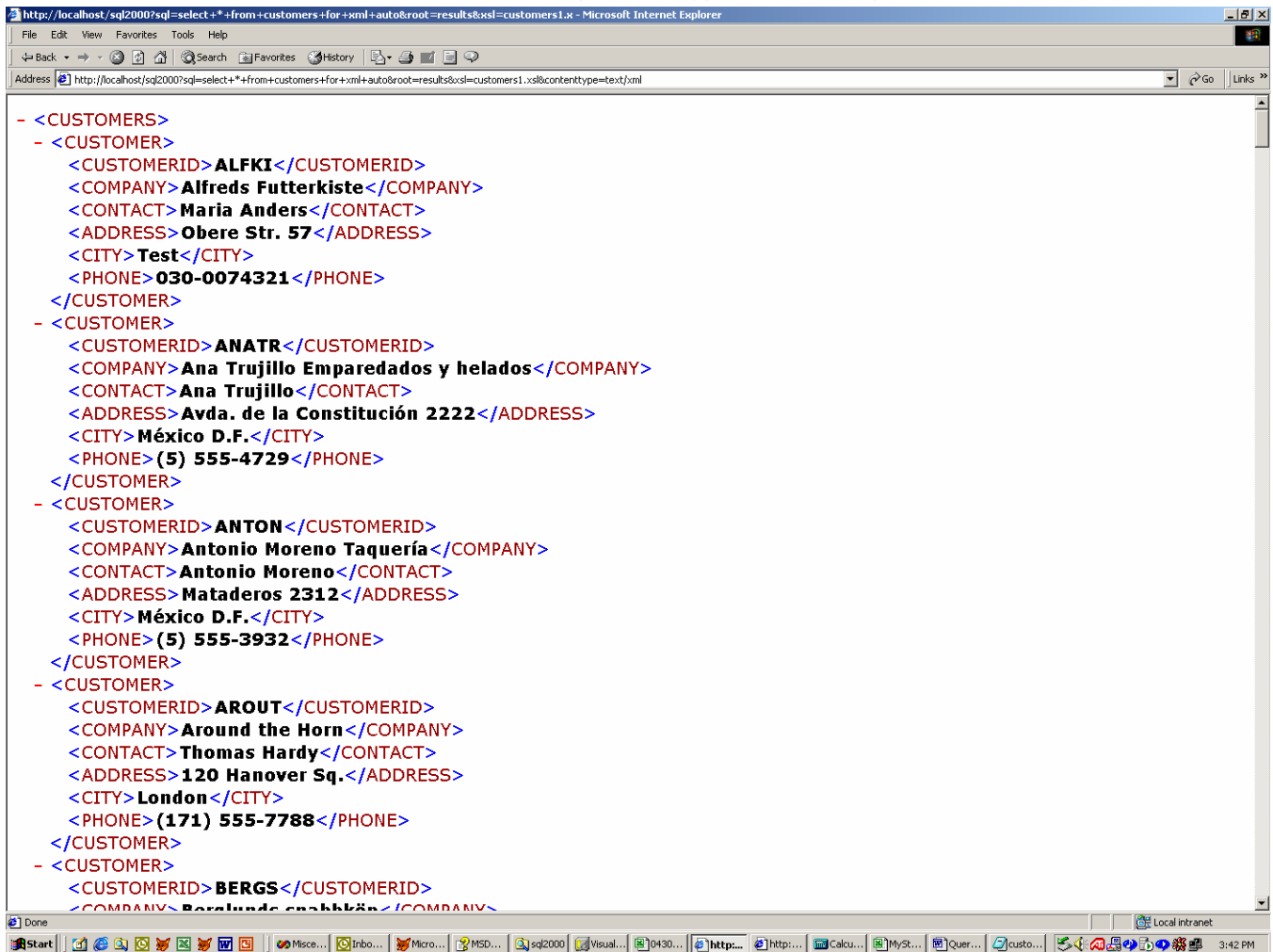
</xsl:template>
</xsl:stylesheet>
```

When the URL query:

```
http://localhost/sql2000?sql=select+*+from+customers+for+xml+auto&root=re
sults&xsl=customers1.xsl&contenttype=text/xml
```

is executed, the style sheet in Listing 1 is applied to the results and a different grammar of XML is returned as shown in Figure 11.

Figure 11. The results of a URL query after the style sheet in Listing 1 is applied.



Often times, however, it becomes necessary to present the results to an end-user in a "grid" or "list" fashion. Using the following style sheet in Listing 2, coupled with the **contenttype** parameter, allows the browser to display the result set as HTML.

Listing 2. Customers2.xsl creates an HTML table.

```

<?xml version="1.0"?>

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
  <xsl:template match="/">

<TABLE width="100%">
  <TR bgcolor="moccasin">
    <TD valign="top"><b>Customer Id</b></TD>
    <TD valign="top"><b>Company</b></TD>
    <TD valign="top"><b>Contact</b></TD>
    <TD valign="top"><b>Address</b></TD>
    <TD valign="top"><b>City</b></TD>

```

```
<TD valign="top"><b>Phone</b></TD>
</TR>

<xsl:for-each select="Results/Customers">

    <TR bgcolor="white">
        <TD valign="top"><xsl:value-of
select="@CustomerID"/></TD>
        <TD valign="top"><xsl:value-of
select="@CompanyName"/></TD>
        <TD valign="top"><xsl:value-of
select="@ContactName"/></TD>
        <TD valign="top"><xsl:value-of
select="@Address"/></TD>
        <TD valign="top"><xsl:value-of
select="@City"/></TD>
        <TD valign="top"><xsl:value-of
select="@Phone"/></TD>
    </TR>

</xsl:for-each>
</TABLE>

</xsl:template>
</xsl:stylesheet>
```

By specifying that the **contenttype** should be **text/html**, the browser will interpret the resulting XML as HTML and display the results appropriately as indicated in Figure 12.

Figure 12. The results of a URL query after applying the style sheet in Listing 2.

http://localhost/sql2000/?sql=select+*+from+customers+for+xml+auto&root=results&xml=customers2.xsl&contentType=text/html - Microsoft Internet Explorer

Address http://localhost/sql2000/?sql=select+*+from+customers+for+xml+auto&root=results&xml=customers2.xsl&contentType=text/html

Customer Id	Company	Contact	Address	City	Phone
ALFKI	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Test	030-0074321
ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	(5) 555-4729
ANTON	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	(5) 555-3932
AROUT	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	(171) 555-7788
BERGS	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	0921-12 34 65
BLAUS	Blauer See Delikatessen	Hanna Moos	Forsterstr. 57	Mannheim	0621-08460
BLONP	Blondesdls père et fils	Frédérique Citeaux	24, place Kléber	Strasbourg	88.60.15.31
BOLID	Bólido Comidas preparadas	Martin Sommer	C/ Araquil, 67	Madrid	(91) 555 22 82
BONAP	Bon app'	Laurence Lebihan	12, rue des Bouchers	Marseille	91.24.45.40
BOTTM	Bottom-Dollar Markets	Elizabeth Lincoln	23 Tsawassen Blvd.	Tsawassen	(604) 555-4729
BSBEV	B's Beverages	Victoria Ashworth	Fauntleroy Circus	London	(171) 555-1212
CACTU	Cactus Comidas para llevar	Patricio Simpson	Cerrito 333	Buenos Aires	(1) 135-5555
CENTC	Centro comercial Moctezuma	Francisco Chang	Sierras de Granada 9993	México D.F.	(5) 555-3392
CHOPS	Chop-suey Chinese	Yang Wang	Hauptstr. 29	Bern	0452-076545
COMMI	Comércio Mineiro	Pedro Afonso	Av. dos Lusíadas, 23	Sao Paulo	(11) 555-7647
CONSH	Consolidated Holdings	Elizabeth Brown	Berkeley Gardens 12 Brewery	London	(171) 555-2282
DRACD	Drachenblut Delikatessen	Sven Ottlieb	Walsertweg 21	Aachen	0241-039123
DUMON	Du monde entier	Janine Labrune	67, rue des Cinquante Otages	Nantes	40.67.88.88
EASTC	Eastern Connection	Ann Devon	35 King George	London	(171) 555-0297
ERNSH	Ernst Handel	Roland Mendel	Kirchgasse 6	Graz	7675-3425
FAMIA	Familia Arquibaldo	Aria Cruz	Rua Orós, 92	Sao Paulo	(11) 555-9857
FISSA	FISSA Fabrica Inter. Salchichas S.A.	Diego Roel	C/ Moralzarzal, 86	Madrid	(91) 555 94 44
FOLIG	Folies gourmandes	Martine Rancé	184, chaussée de Tournai	Lille	20.16.10.16
FOLKO	Folk och få HB	Maria Larsson	Åkergatan 24	Bräcke	0695-34 67 21
FRANK	Frankenversand	Peter Franken	Berliner Platz 43	München	089-0877310
FRANR	France restauration	Carine Schmitt	54, rue Royale	Nantes	40.32.21.21
FRANS	Franchi S.p.A.	Paolo Accorti	Via Monte Bianco 34	Torino	011-4988260
FURIB	Furia Bacalhau e Frutos do Mar	Lino Rodriguez	Jardim das rosas n. 32	Lisboa	(1) 354-2534
GALED	Galeria del gastrónomo	Eduardo Saavedra	Rambla de Cataluña, 23	Barcelona	(93) 203 4560
GODOS	Godos Cocina Típica	José Pedro Freyre	C/ Romero, 33	Sevilla	(95) 555 82 82
GOURL	Gourmet Lanchonetes	André Fonseca	Av. Brasil, 442	Campinas	(11) 555-9482
GREAL	Great Lakes Food Market	Howard Snyder	2732 Baker Blvd.	Eugene	(503) 555-7555

Template Queries

Another method of retrieving an XML result set from SQL Server 2000, is to use what's called Template queries. These are XML files that tell SQL Server how to run queries, what the root node will be, etc. These files eliminate the need to specify the select statement at the URL level. Revisiting our query:

```
SELECT * FROM Customers
```

as a Template query would appear as shown in Listing 3, below.

Listing 3. Customers1.xml lists a Template query to select all customers in Northwind.

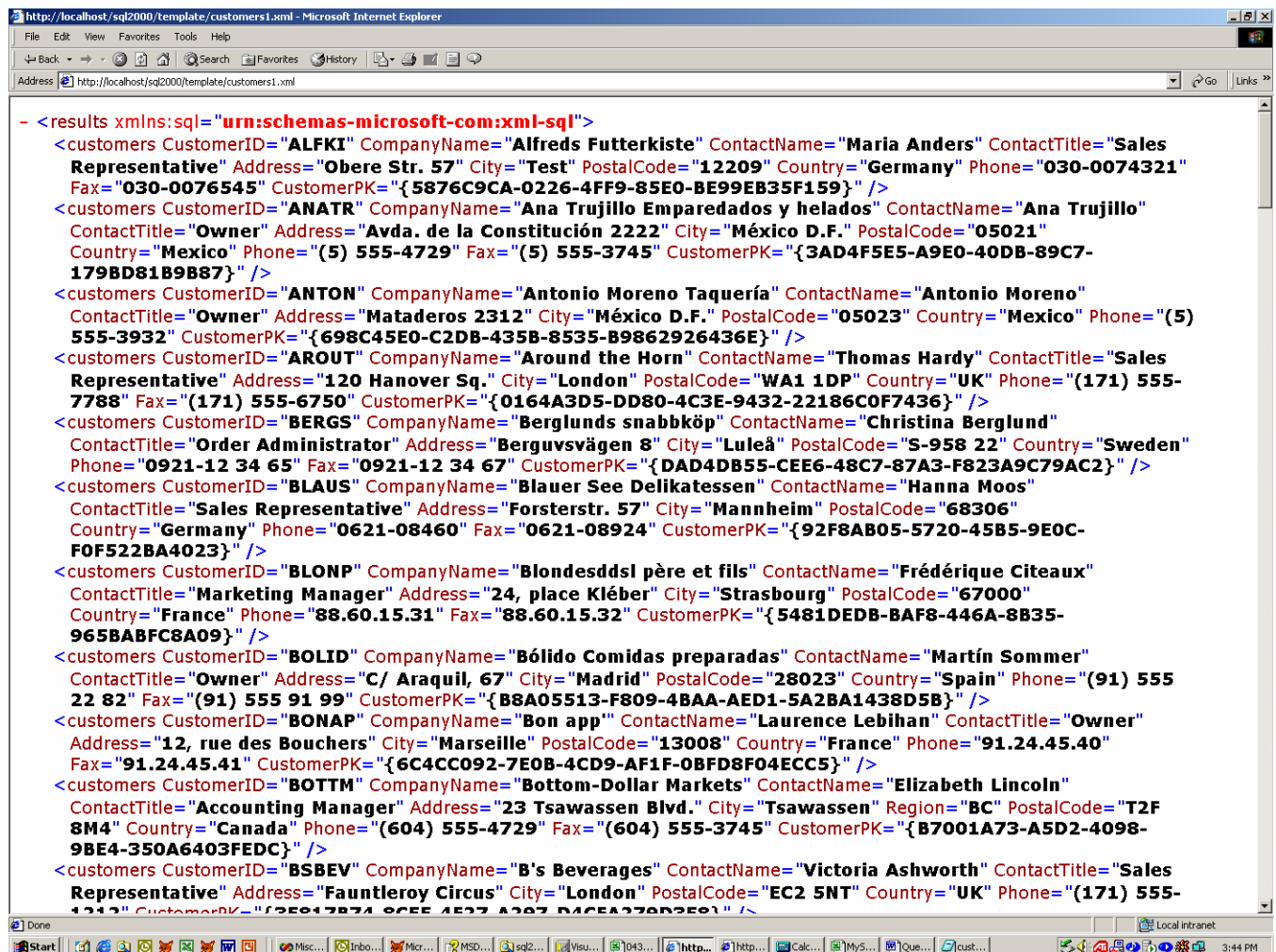
```
<results xmlns:sql="urn:schemas-microsoft-com:xml-sql">
  <sql:query>
    select * from customers for xml auto
  </sql:query>
</results>
```


To run this query, save the above template as Customers1.xml and store it in the Template virtual directory and simply navigate to:

<http://localhost/sql2000/template/customers1.xml>

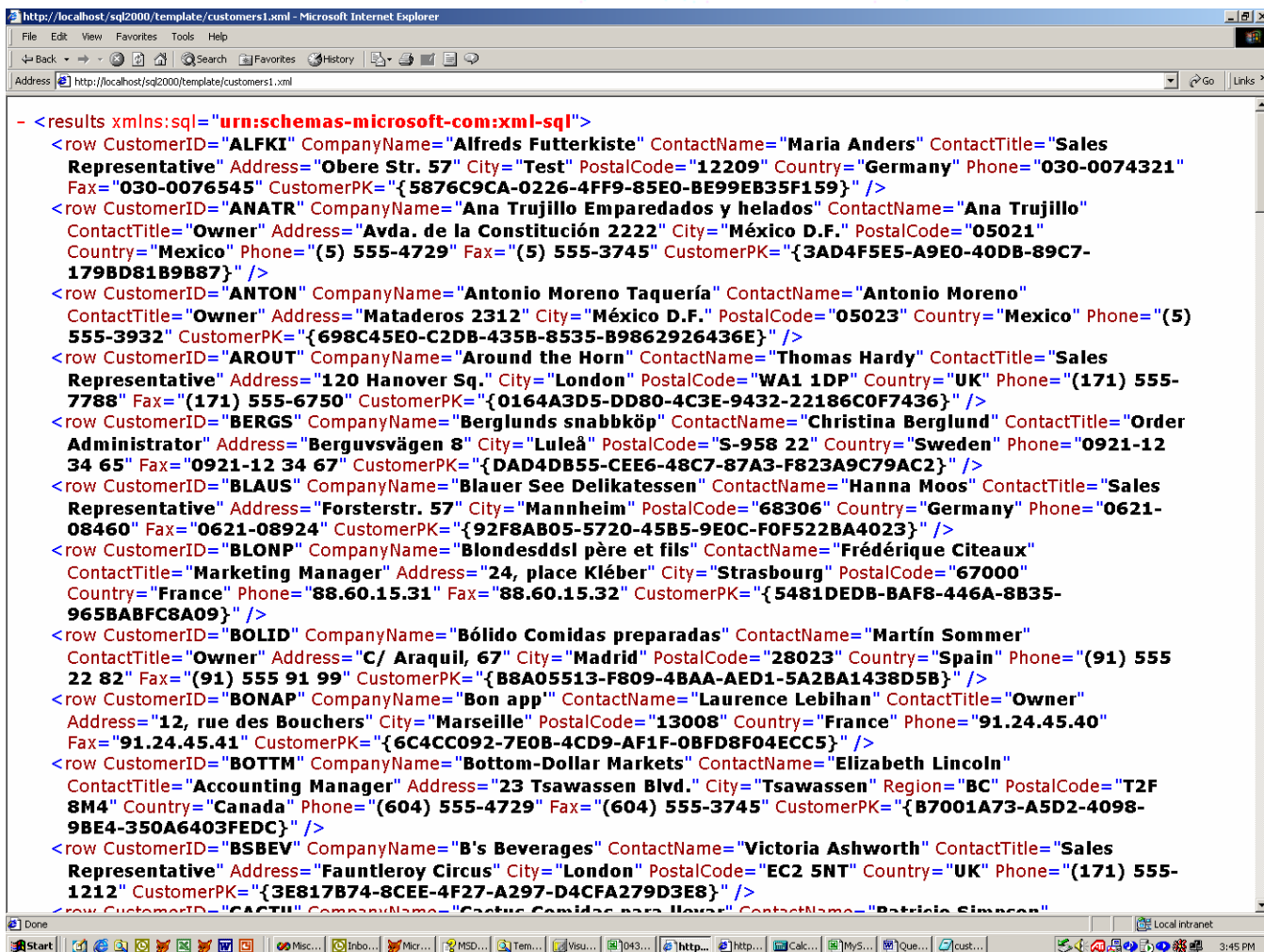
which returns the XML in the native SQL Server format shown in Figure 13.

Figure 13. The results of navigating to Customers1.xml.



Replacing the keyword **AUTO** with **RAW** in the above Template query will return the results shown in Figure 14.

Figure 14. The results of navigating to the revised Customers1.xml.



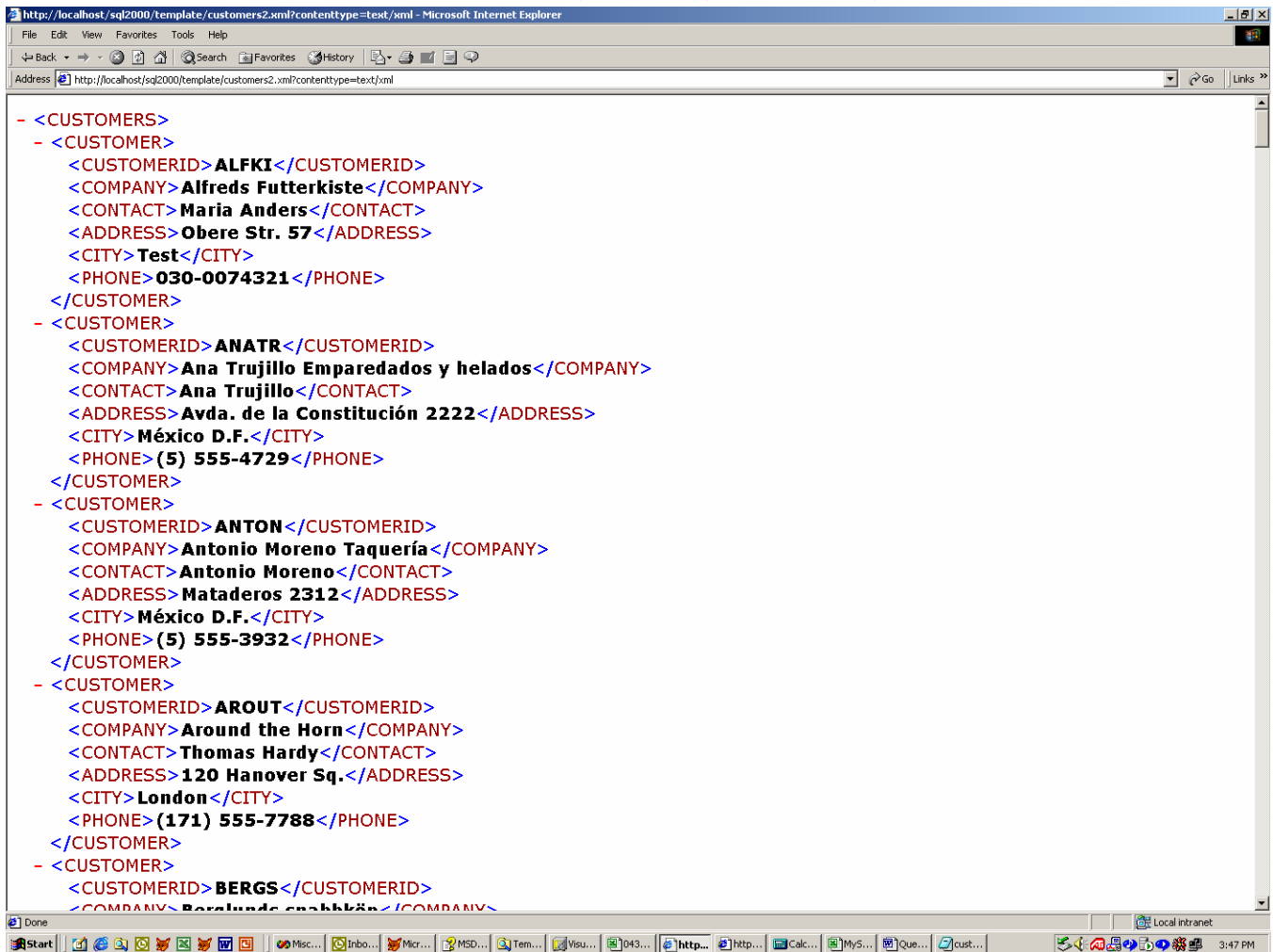
To apply a style sheet to the result set, the Template query would appear as shown in Listing 4.

Listing 4. Customers2.xml lists a Template query that uses a style sheet.

```
<results xmlns:sql="urn:schemas-microsoft-com:xml-sql"
sql:xsl='../customers1.xsl'>
  <sql:query>
    select * from customers for xml auto
  </sql:query>
</results>
```

which, when combined with the **contenttype** parameter, will return the results in Figure 15.

Figure 15. The results of navigating to the Template shown in Listing 4.



As with URL queries, you can format the final results to be displayed in HTML. In this case, simply change the customers1.xsl filename to customers2.xsl. The results will be in an HTML table as shown in Figure 16.

Figure 16. The results of changing the style sheet to Customers2.xsl.

http://localhost/sql2000/template/customers3.xml - Microsoft Internet Explorer

Address http://localhost/sql2000/template/customers3.xml

Customer Id	Company	Contact	Address	City	Phone
ALFKI	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Test	030-0074321
ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	(5) 555-4729
ANTON	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	(5) 555-3932
AROUT	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	(171) 555-7788
BERGS	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	0921-12 34 65
BLAUS	Blauer See Delikatessen	Hanna Moos	Forsterstr. 57	Mannheim	0621-08460
BLONP	Blondesdls père et fils	Frédérique Citeaux	24, place Kléber	Strasbourg	88.60.15.31
BOLID	Bólido Comidas preparadas	Martin Sommer	C/ Araquil, 67	Madrid	(91) 555 22 82
BONAP	Bon app'	Laurence Lebihan	12, rue des Bouchers	Marseille	91.24.45.40
BOTTM	Bottom-Dollar Markets	Elizabeth Lincoln	23 Tsawassen Blvd.	Tsawassen	(604) 555-4729
BSBEV	B's Beverages	Victoria Ashworth	Fauntleroy Circus	London	(171) 555-1212
CACTU	Cactus Comidas para llevar	Patricio Simpson	Cerrito 333	Buenos Aires	(1) 135-5555
CENTC	Centro comercial Moctezuma	Francisco Chang	Sierras de Granada 9993	México D.F.	(5) 555-3392
CHOPS	Chop-suey Chinese	Yang Wang	Hauptstr. 29	Bern	0452-076545
COMMI	Comércio Mineiro	Pedro Afonso	Av. dos Lusíadas, 23	Sao Paulo	(11) 555-7647
CONSH	Consolidated Holdings	Elizabeth Brown	Berkeley Gardens 12 Brewery	London	(171) 555-2282
DRACD	Drachenblut Delikatessen	Sven Ottlieb	Walserweg 21	Aachen	0241-039123
DUMON	Du monde entier	Janine Labrune	67, rue des Cinquante Otages	Nantes	40.67.88.88
EASTC	Eastern Connection	Ann Devon	35 King George	London	(171) 555-0297
ERNSH	Ernst Handel	Roland Mendel	Kirchgasse 6	Graz	7675-3425
FAMIA	Familia Arquibaldo	Aria Cruz	Rua Orós, 92	Sao Paulo	(11) 555-9857
FISSA	FISSA Fabrica Inter. Salchichas S.A.	Diego Roel	C/ Moralzarzal, 86	Madrid	(91) 555 94 44
FOLIG	Folies gourmandes	Martine Rancé	184, chaussée de Tournai	Lille	20.16.10.16
FOLKO	Folk och få HB	Maria Larsson	Åkergatan 24	Bräcke	0695-34 67 21
FRANK	Frankenversand	Peter Franken	Berliner Platz 43	München	089-0877310
FRANR	France restauration	Carine Schmitt	54, rue Royale	Nantes	40.32.21.21
FRANS	Franchi S.p.A.	Paolo Accorti	Via Monte Bianco 34	Torino	011-4988260
FURIB	Furia Bacalhau e Frutos do Mar	Lino Rodriguez	Jardim das rosas n. 32	Lisboa	(1) 354-2534
GALED	Galeria del gastrónomo	Eduardo Saavedra	Rambla de Cataluña, 23	Barcelona	(93) 203 4560
GODOS	Godos Cocina Típica	José Pedro Freyre	C/ Romero, 33	Sevilla	(95) 555 82 82
GOURL	Gourmet Lanchonetes	André Fonseca	Av. Brasil, 442	Campinas	(11) 555-9482
GREAL	Great Lakes Food Market	Howard Snyder	2732 Baker Blvd.	Eugene	(503) 555-7555

Template queries also have the potential to accept parameters to help filter the result set. In this case, the Template file will appear as shown in Listing 5.

Listing 5. Customers4.xml lists a Template query that accepts parameters.

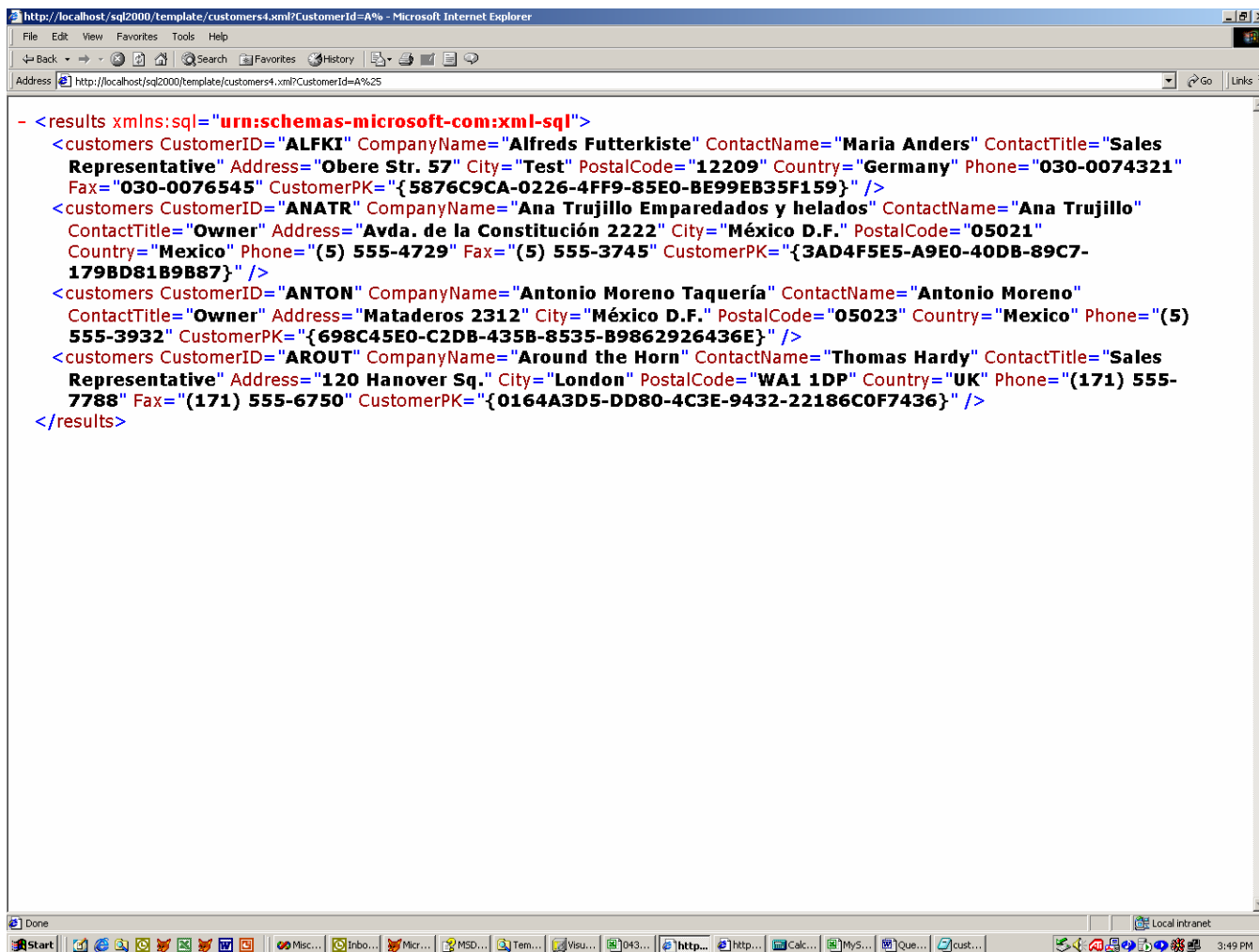
```
<results xmlns:sql="urn:schemas-microsoft-com:xml-sql"
sql:xsl='../customers1.xsl'>
  <sql:header>
    <sql:param name='CustomerId'>%</sql:param>
  </sql:header>
  <sql:query>
    select * from customers where customerid like @CustomerId for xml auto
  </sql:query>
</results>
```

In this example, we specify that this Template query will accept one parameter: CustomerId, and that this parameter has a default value of %, which in conjunction with the query, will return all records from the customers table. To execute this query, simply navigate to:

<http://localhost/sql2000/template/customers4.xml?CustomerId=A%25>

This will return only those records from the customers table where the CustomerId field begins with an A as shown in Figure 17.

Figure 17. The results of navigating to Customers4.xml.



Dynamic Template Queries

Although it may be easier to create a Template query file on the server to execute a query, SQL Server 2000 allows the content of the template to be specified dynamically on the URL, by using the "template" parameter. This allows template queries to be created and executed dynamically.

Defining XPath Queries within Template Files

Although you may access the XPath query directly from the URL, you can also use an XPath query within a Template query file.

XPath Queries

Imagine the potential of accessing your database without needing to any specifics about how that data is stored. Imagine the possibilities of allowing the DBA the freedom to change the database structure without affecting your software code. Impossible? Not with XPath Queries and XDR Schemas. Using these two methods with each other allow practically any developer to write a program to access data in SQL Server 2000 just by knowing the structure of the XML documents returned from SQL Server 2000.

While both Template Queries and URL Queries allow the developer to retrieve XML data back from SQL Server 2000, they do require knowledge of SQL Select statements, stored procedures, and the structure of the database. However, by using XPath Queries in conjunction with XDR Schemas, these knowledge requirements can be removed.

XML – Data Reduced (XDR) Schemas define the structure of the XML document returned from SQL Server 2000 and enables various constraints to be placed upon the data returned. Unlike a Document Type Definition (DTD), an XML Schema describes the structure of an XML document using an XML grammar.

Authoring Annotated XSD Schemas

An XML – Data Reduced Schema is simply an XML Schema with specific attributes used in defining the XML elements and attributes. Every XDR Schema must include the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:sql="urn:schemas-microsoft-com:mapping-schema">
...
</xsd:schema>
```

Notice, that the above example is an XML document with a root node of schema. This may or may not make sense to you. Remember that an XML Schema defines the structure and data constraints of an XML document using an XML grammar.

There are 2 basic attributes and 1 basic element needed to author XSD Schemas. The attributes needed are **sql:field** and **sql:relation**. The element is the **sql:relationship** element. The **sql:relation** attribute is used to map an element to a table. This has the effect of creating 1 XML element for every record in the table. The **sql:field** attribute is used to map a particular attribute or node value to a field from the related table. The **sql:relationship** element is used to relate elements within the XML document to other elements. It defines the 2 tables and the join condition necessary to relate them together.

Using those XSD attributes and elements, an XSD schema can be authored to return data from SQL Server 2000 in a specific format. The only required XSD attribute is the **sql:relation** attribute. This attribute refers to a table or view in the database and can be placed on an **ElementType**, element, or attribute element in the XSD Schema. The following schema shown in Listing 6 is a simple example of using an **sql:relation** attribute in an XSD Schema for the Customer table in the Northwind database.

Listing 6. Customers1.xsd lists a simple annotated XSD schema.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:sql="urn:schemas-microsoft-com:mapping-schema">

  <xsd:element name="CUSTOMER" sql:relation="Customers">
    <xsd:complexType>
      <xsd:attribute name="CustomerID" type="xsd:string"/>
      <xsd:attribute name="CompanyName" type="xsd:string"/>
      <xsd:attribute name="ContactName" type="xsd:string"/>
      <xsd:attribute name="Address" type="xsd:string"/>
      <xsd:attribute name="City" type="xsd:string"/>
      <xsd:attribute name="Phone" type="xsd:string"/>
    </xsd:complexType>
  </xsd:element>

</xsd:schema>
```

By specifying the **sql:relation** attribute on an **ElementType** element within our schema, the relation is inherited by all elements and attributes of that **ElementType**. This means we do not need to specify the **sql:relation** attribute on every element or attribute element within our schema. Because we defined our attribute names in the exact manner they exist in the Northwind database, we did not need to use the **sql:field** attribute. Keep in mind, that XML is case – sensitive. Therefore, the attribute names defined above must match exactly with the field names defined in the database for this XSD Schema to work. This schema returns an XML document whose structure matches that of the one defined in the schema, as shown below in Figure 18.

Figure 18. The results of an XPath query performed against Customers1.xsd.

```

http://localhost/sql2000/schema/sampleschema1.xml/CUSTOMER?root=ROOT - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Search Favorites History
Address http://localhost/sql2000/schema/sampleschema1.xml/CUSTOMER?root=ROOT

<?xml version="1.0" encoding="utf-8" ?>
- <ROOT>
  <CUSTOMER CustomerID="ALFKI" CompanyName="Alfreds Futterkiste" ContactName="Maria Anders" Address="Obere Str.
  57" City="Test" Phone="030-0074321" />
  <CUSTOMER CustomerID="ANATR" CompanyName="Ana Trujillo Emparedados y helados" ContactName="Ana Trujillo"
  Address="Avda. de la Constitución 2222" City="México D.F." Phone="(5) 555-4729" />
  <CUSTOMER CustomerID="ANTON" CompanyName="Antonio Moreno Taquería" ContactName="Antonio Moreno"
  Address="Mataderos 2312" City="México D.F." Phone="(5) 555-3932" />
  <CUSTOMER CustomerID="AROUT" CompanyName="Around the Horn" ContactName="Thomas Hardy" Address="120
  Hanover Sq." City="London" Phone="(171) 555-7788" />
  <CUSTOMER CustomerID="BERGS" CompanyName="Berglunds snabbköp" ContactName="Christina Berglund"
  Address="Berguvsvägen 8" City="Luleå" Phone="0921-12 34 65" />
  <CUSTOMER CustomerID="BLAUS" CompanyName="Blauer See Delikatessen" ContactName="Hanna Moos"
  Address="Forsterstr. 57" City="Mannheim" Phone="0621-08460" />
  <CUSTOMER CustomerID="BLONP" CompanyName="Blondesddsl père et fils" ContactName="Frédérique Citeaux"
  Address="24, place Kléber" City="Strasbourg" Phone="88.60.15.31" />
  <CUSTOMER CustomerID="BOLID" CompanyName="Bólido Comidas preparadas" ContactName="Martín Sommer"
  Address="C/ Araquil, 67" City="Madrid" Phone="(91) 555 22 82" />
  <CUSTOMER CustomerID="BONAP" CompanyName="Bon app" ContactName="Laurence Lebihan" Address="12, rue des
  Bouchers" City="Marseille" Phone="91.24.45.40" />
  <CUSTOMER CustomerID="BOTTM" CompanyName="Bottom-Dollar Markets" ContactName="Elizabeth Lincoln"
  Address="23 Tsawassen Blvd." City="Tsawassen" Phone="(604) 555-4729" />
  <CUSTOMER CustomerID="BSBEV" CompanyName="B's Beverages" ContactName="Victoria Ashworth" Address="Fautleroy
  Circus" City="London" Phone="(171) 555-1212" />
  <CUSTOMER CustomerID="CACTU" CompanyName="Cactus Comidas para llevar" ContactName="Patricio Simpson"
  Address="Cerrito 333" City="Buenos Aires" Phone="(1) 135-5555" />
  <CUSTOMER CustomerID="CENTC" CompanyName="Centro comercial Moctezuma" ContactName="Francisco Chang"
  Address="Sierras de Granada 9993" City="México D.F." Phone="(5) 555-3392" />
  <CUSTOMER CustomerID="CHOPS" CompanyName="Chop-suey Chinese" ContactName="Yang Wang" Address="Hauptstr.
  29" City="Bern" Phone="0452-076545" />
  <CUSTOMER CustomerID="COMMI" CompanyName="Comércio Mineiro" ContactName="Pedro Afonso" Address="Av. dos
  Lusíadas, 23" City="Sao Paulo" Phone="(11) 555-7647" />
  <CUSTOMER CustomerID="CONSH" CompanyName="Consolidated Holdings" ContactName="Elizabeth Brown"
  Address="Berkeley Gardens 12 Brewery" City="London" Phone="(171) 555-2282" />
  <CUSTOMER CustomerID="DRACD" CompanyName="Drachenblut Delikatessen" ContactName="Sven Ottlieb"
  Address="Walsertweg 21" City="Aachen" Phone="0241-039123" />
  <CUSTOMER CustomerID="DUMON" CompanyName="Du monde entier" ContactName="Janine Labrune" Address="67, rue
  
```

The **sql:field** attribute may be used in conjunction with the **sql:relation** attribute to create elements or attributes that do not exactly match their definitions in the database. For instance, the following XSD Schema in Listing 7 will produce an XML document with attributes for the fields in the Customer table.

Listing 7. Customers2.xsd lists an example of mapping attributes to fields in the database.

```

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:sql="urn:schemas-microsoft-com:mapping-schema"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:element name="CUSTOMER" sql:relation="Customers">
    <xsd:complexType>
      <xsd:attribute name="Id" type="xsd:string" sql:field="CustomerID"/>
      <xsd:attribute name="Company" type="xsd:string"
        sql:field="CompanyName"/>
      <xsd:attribute name="Contact" type="xsd:string"
        sql:field="ContactName"/>
    
```



```

<xsd:attribute name="Address" type="xsd:string"/>
<xsd:attribute name="City" type="xsd:string"/>
<xsd:attribute name="Phone" type="xsd:string"/>
</xsd:complexType>
</xsd:element>

</xsd:schema>

```

Performing an XPath query against the schema in Listing 7 will produce the results shown in Figure 19.

Figure 19. The results of performing an XPath query against Customers2.xsd.



Now let's say we don't like having the fields mapped to attributes. Instead, we could use the following schema listed in Listing 8 to produce an XML document with elements for the fields in the Customer table.

Listing 8. Customers3.xsd shows an example of mapping elements to fields.

```

<?xml version="1.0" encoding="UTF-8"?>

```

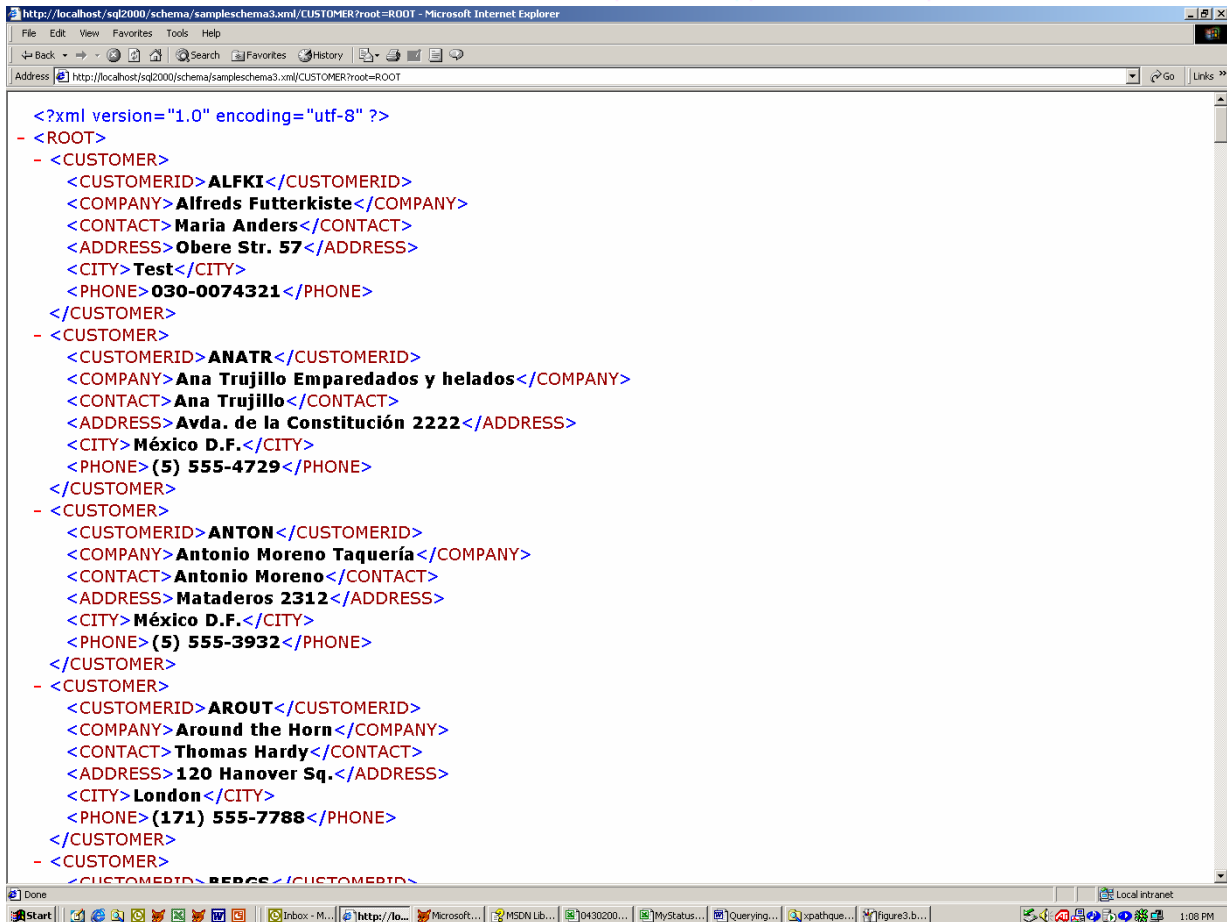
```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:sql="urn:schemas-microsoft-com:mapping-schema">

  <xsd:element name="CUSTOMER" sql:relation="Customers">
    <xsd:complexType>
      <xsd:all>
        <xsd:element name="CUSTOMERID" type="xsd:string"
sql:field="CustomerID"/>
        <xsd:element name="COMPANY" type="xsd:string"
sql:field="CompanyName"/>
        <xsd:element name="CONTACT" type="xsd:string"
sql:field="ContactName"/>
        <xsd:element name="ADDRESS" type="xsd:string" sql:field="Address"/>
        <xsd:element name="CITY" type="xsd:string" sql:field="City"/>
        <xsd:element name="PHONE" type="xsd:string" sql:field="Phone"/>
      </xsd:all>
    </xsd:complexType>
  </xsd:element>

</xsd:schema>
```

Navigating to the schema in Listing 8 returns the results shown in Figure 20.

Figure 20. The results of performing an XPath query against Customers3.xsd.



In addition to using the **sql:relation** and **sql:field** attributes, you can use the **sql:relationship** element to produce nested XML documents where elements may contain related child elements. A sample XSD Schema for this follows in Listing 9 below.

Listing 9. Customers4.xsd demonstrates how to define relationships between elements.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:sql="urn:schemas-microsoft-com:mapping-schema"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:element name="CUSTOMER" sql:relation="Customers">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="CUSTOMERID" type="xsd:string"
          sql:field="CustomerID"/>
        <xsd:element name="COMPANY" type="xsd:string"
          sql:field="CompanyName"/>
        <xsd:element name="CONTACT" type="xsd:string"
          sql:field="ContactName"/>
        <xsd:element name="ADDRESS" type="xsd:string" sql:field="Address"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

```

<xsd:element name="CITY" type="xsd:string" sql:field="City"/>
<xsd:element name="PHONE" type="xsd:string" sql:field="Phone"/>

  <xsd:element name="ORDER" maxOccurs="unbounded"
sql:relation="Orders">
  <xsd:annotation>
  <xsd:appinfo>
    <sql:relationship parent="Customers" parent-key="CustomerID"
child="Orders" child-key="CustomerID"/>
  </xsd:appinfo>
  </xsd:annotation>

  <xsd:complexType>
  <xsd:sequence>
    <xsd:element name="ORDERID" type="xsd:integer"
sql:field="OrderID"/>
    <xsd:element name="ORDERDATE" type="xsd:date"
sql:field="OrderDate"/>

    <xsd:element name="DETAILS" maxOccurs="unbounded"
sql:relation="[Order Details]">
    <xsd:annotation>
    <xsd:appinfo>
      <sql:relationship parent="Orders" parent-key="OrderID"
child="[Order Details]" child-key="OrderID"/>
    </xsd:appinfo>
    </xsd:annotation>

    <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="ORDERID" type="xsd:integer"
sql:field="OrderID"/>
      <xsd:element name="PRODUCTID" type="xsd:integer"
sql:field="ProductID"/>
      <xsd:element name="UNITPRICE" sql:field="UnitPrice">
      <xsd:simpleType>
      <xsd:restriction base="xsd:decimal">
      <xsd:fractionDigits value="2"/>
      </xsd:restriction>
      </xsd:simpleType>
      </xsd:element>
      <xsd:element name="QUANTITY" type="xsd:positiveInteger"
sql:field="Quantity"/>
    </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  </xsd:sequence>
</xsd:complexType>
</xsd:element>

</xsd:sequence>

```

```

</xsd:complexType>
</xsd:element>

</xsd:schema>

```

Performing an XPath query against the schema shown in Listing 9 produces the following output shown in Figure 21.

Figure 21. The results of performing an XPath query against Customers4.xsd.

```

<?xml version="1.0" encoding="utf-8" ?>
- <ROOT>
- <CUSTOMER>
  <CUSTOMERID>ALFKI</CUSTOMERID>
  <COMPANY>Alfreds Futterkiste</COMPANY>
  <CONTACT>Maria Anders</CONTACT>
  <ADDRESS>Obere Str. 57</ADDRESS>
  <CITY>Test</CITY>
  <PHONE>030-0074321</PHONE>
- <ORDER>
  <ORDERID>10643</ORDERID>
  <ORDERDATE>1997-08-25T00:00:00</ORDERDATE>
</ORDER>
- <ORDER>
  <ORDERID>10692</ORDERID>
  <ORDERDATE>1997-10-03T00:00:00</ORDERDATE>
</ORDER>
- <ORDER>
  <ORDERID>10702</ORDERID>
  <ORDERDATE>1997-10-13T00:00:00</ORDERDATE>
</ORDER>
- <ORDER>
  <ORDERID>10835</ORDERID>
  <ORDERDATE>1998-01-15T00:00:00</ORDERDATE>
</ORDER>
- <ORDER>
  <ORDERID>10952</ORDERID>
  <ORDERDATE>1998-03-16T00:00:00</ORDERDATE>
</ORDER>
- <ORDER>
  <ORDERID>11011</ORDERID>
  <ORDERDATE>1998-04-09T00:00:00</ORDERDATE>
</ORDER>
</CUSTOMER>
- <CUSTOMER>
  <CUSTOMERID>ANATR</CUSTOMERID>

```

There are 4 attributes on the **sql:relationship** element that must be specified: **key**, **key-relation**, **foreign-key**, **foreign-relation**. The **key-relation** attribute specifies the Parent Table and the **key** attribute specifies the key on the Parent Table to use to relate it to a child table. The **foreign-relation** attribute specifies the Child Table and the **foreign-key** attribute specifies the key on the Child Table used to relate it to the Parent Table. SQL Server 2000 is able to relate and nest related XML elements using the values specified in the attributes of the **sql:relationship** element.

XPath Syntax

XPath Queries, coupled with XDR Schemas, allow the developer to access and query data from SQL Server 2000 in the same manner as using the XMLDOM object. For instance, navigating to the following URL:

```
http://localhost/sql2000/schema/customers1.xsd/CUSTOMER?root=ROOT
```

produces an XML document without writing a single line of an SQL Select statement.

Similarly, navigating to:

```
http://localhost/sql2000/schema/customers4.xsd/CUSTOMER?root=ROOT
```

produces an XML document with customers and their related orders.

Keep in mind that you may use the XMLDOM to query SQL Server 2000. For instance, you could do the following:

```
Dim loXML As Object
SET loXML = CreateObject("MSXML2.DOMDocument")
loXML.Load("http://localhost/sql2000/schema/customers4.xsd/CUSTOMER?root=ROOT")
```

The above example will load every customer with his or her appropriate orders into the XMLDOM object. However, rather than download the entire set of data into the XMLDOM object and then query its contents for specific nodes, why not let SQL Server 2000 do that for you automatically on the server and send back a reduced data set? Let's say you only wanted customer ALFKI. In this scenario, simply navigate to:

```
http://localhost/sql2000/schema/customers4.xsd/CUSTOMER[CUSTOMERID='ALFKI']?root=ROOT.
```

The above URL will produce the following results shown in Figure 22.

Figure 22. The results of navigating to the above URL.



```

<?xml version="1.0" encoding="utf-8" ?>
- <ROOT.>
  - <CUSTOMER>
    <CUSTOMERID>ALFKI</CUSTOMERID>
    <COMPANY>Alfreds Futterkiste</COMPANY>
    <CONTACT>Maria Anders</CONTACT>
    <ADDRESS>Obere Str. 57</ADDRESS>
    <CITY>Test</CITY>
    <PHONE>030-0074321</PHONE>
  - <ORDER>
    <ORDERID>10643</ORDERID>
    <ORDERDATE>1997-08-25T00:00:00</ORDERDATE>
  </ORDER>
  - <ORDER>
    <ORDERID>10692</ORDERID>
    <ORDERDATE>1997-10-03T00:00:00</ORDERDATE>
  </ORDER>
  - <ORDER>
    <ORDERID>10702</ORDERID>
    <ORDERDATE>1997-10-13T00:00:00</ORDERDATE>
  </ORDER>
  - <ORDER>
    <ORDERID>10835</ORDERID>
    <ORDERDATE>1998-01-15T00:00:00</ORDERDATE>
  </ORDER>
  - <ORDER>
    <ORDERID>10952</ORDERID>
    <ORDERDATE>1998-03-16T00:00:00</ORDERDATE>
  </ORDER>
  - <ORDER>
    <ORDERID>11011</ORDERID>
    <ORDERDATE>1998-04-09T00:00:00</ORDERDATE>
  </ORDER>
</CUSTOMER>
</ROOT.>

```

Notice that the syntax for an XPath query is almost identical to the syntax used for an XSL Pattern match within the XMLDOM itself. In truth, the main difference is the exclusion of the root node in the XPath query that would otherwise have to be included in an XSL Pattern match in the XMLDOM.

Alternative Ways to Query SQL Server 2000

Although this paper concentrates on using Internet Explorer to query data from SQL Server 2000 in XML format, Internet Explorer is not the only tool that can be used. Microsoft's **XMLDOM** object can load an XML file via a URL. This can provide an alternative method of accessing SQL Server data in XML format without having to use Internet Explorer. Since loading information from a URL may take a little longer than loading a file from a hard drive, it is important that you set the **ASync** property of the **DOMDocument** object to false. Another alternative available with .NET are the classes contained within the System.Xml namespace, or even the DataSet class in the System.Data namespace.



Summary

XML is quickly becoming the preferred method of passing information, not only for the Internet, but also across applications, and even within the same application. Until now, developers have been forced to create their own routines to automate messaging and to convert data contained within a database into XML.

Now, with SQL Server 2000, much of those these tasks can be handled in a much more efficient manner. That leaves the developer with more time to perform the more important tasks of actual programming, by not having to worry about writing conversion routines to convert relational data into XML. What else can be said, except that the future looks bright.