José Antônio da Cunha

IFRN

Disciplina:ADM Banco de Dados

Desenvolvendo com XML

Tópicos:

- O tipo de dados XML
- O suporte XQuery
- Os métodos Data Type XML
- Índices XML
- Usando a cláusula For XML
- OPENXML
- XML Bulk Load
- Acesso nativo SOAP

O SQL Server 2005 adicionou suporte para muitas características XML. Primeiro, o SQL Server 2005 forneceu um novo nível de armazenamento unificado para XML e dados relacionais, adicionando o novo tipo de dado XML. O tipo de dado nativo do SQL Server 2005 fornece suporte tanto para consultas XML nativas usando XQUERY, como forte tipificação de dados associando o tipo de dado XML a um XSD (Extensible Schema Definition). O suporte XML é intimamente integrado com o mecanismo relacional de banco de dados do SQL Server 2005. O SQL Server 2005 fornece suporte para triggers sobre XML, replicação de dados XML, e carga bruta (bulk load) de dados XML, assim como suporte aprimorado para o acesso a dados via SOAP.

O tipo de dados XML

O tipo de dados XML pode ser usado como uma coluna em uma tabela ou uma variável ou um parâmetro em uma stored procedure. Ele pode ser usado para armazenar tanto dados tipados como não tipados. Se o dado armazenado em uma coluna XML não possuir um esquema XSD, então ele é considerado não tipado. Se houver um esquema associado, então o SQL Server 2005 irá verificar todos os dados inseridos no tipo de dados contra o esquema para garantir que os dados armazenados estão de acordo com a definição do

esquema. O tipo de dados XML pode aceitar um máximo de 2GB de dados, e sua estrutura de armazenamento em disco é similar ao tipo de dados varbinary(MAX).

O código a seguir ilustra a criação de uma simples tabela que usa o novo tipo de dados XML:

```
CREATE DATABASE TesteXML
GO

USE TesteXML
GO

--Criando uma tabela que usa tipo de dados XML
CREATE TABLE MyXMLDoc
(
  DocID int primary key,
  MyDoc XML
)
GO
```

No código acima, a coluna MeuDoc usa o tipo de dados XML para especificar que a coluna irá armazenar os dados XML.

O exemplo a seguir, mostra como você pode armazenar um valor em uma coluna XML usando a declaração T-SQL INSERT:

```
INSERT INTO MyXMLDoc Values (1, '<MyXMLDoc>
                                 <DocumentoID>1</DocumentoID>
                                 <DocumentText>Text</DocumentText>
                                 </MyXMLDoc>')
```

Validação de dados usando um esquema XSD

O tipo de dados nativo XML verifica a garantia de que quaisquer dados armazenados em uma variável ou coluna XML, seja um documento XML válido. Por si só, ela não verifica mais do que isso. Entretanto, a Microsoft projetou o tipo de dados XML para ser capaz de suportar a validação de documentos mais sofisticados usando um esquema XSD. Quando um esquema XSD é definido para uma coluna do tipo XML, o mecanismo do SQL Server irá verificar para garantir que todos os dados armazenados na coluna XML, estão de acordo com a definição fornecida pelo seu esquema XSD.

Criando o esquema XSD

O código a seguir mostra um esquema XSD para o documento XML simples que foi usado no exemplo anterior.

```
<?xml version="1.0" encoding="utf-16"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
             elementFormDefault="qualified"
targetNamespace="MyXMLDocSchema"
             xmlns="MyXMLDocSchema">
     <xs:element name="MyXMLDoc">
          <xs:complexType>
               <xs:sequence>
                    <xs:element name="DocumentID"
type="xs:int" />
                    <xs:element name="DocumentBody"
type="xs:string" />
               </xs:sequence>
          </xs:complexType>
     </xs:element>
</xs:schema>
```

Esse esquema XSD usa o namespace MyXMDocSchema e define um document XML que possui um elemento complexo chamado MyXMLDoc. O elemento complexo MyXMLDoc contém dois elementos simples. O primeiro elemento simples deve ser chamado DocumentID, e o segundo elemento simples é chamado DocumentBody. O elemento DocumentID deve conter um inteiro (integer), enquanto o elemento DocumentBody deve conter um tipo de dados XML string.

Para criar uma coluna ou variável XML fortemente tipada, você precisa primeiro registrar o esquema XSD com o SQL Server usando a declaração CREATE XML SCHEMA COLLECTION. Isso registra o esquema no banco de dados SQL Server. Após o esquema XSD ser registrado, ele pode ser usado em um tipo de dados XML. Vamos vê um exemplo:

Uma vez que você tenha registrado o esquema XML com o banco de dados do SQL Server 2005, você poderá associar as variáveis e colunas XML com este esquema. O exemplo a seguir ilustra como você pode criar uma tabela que usa o MyXMLDocSchema que foi criado anteriormente.

```
CREATE TABLE MyXMLDocs
(
  DocID INT PRIMARY KEY,
  MyXMLDOC XML(MyXMLDocSchema)
)
GO
```

Não há alteração na forma básica de inserir dados em uma coluna XML tipada. Entretanto, uma vez que a coluna DocumentBody tenha sido tipada, todos os dados que forem armazenados deverão se adaptar à definição do esquema XSD. O código a seguir mostra como você pode inserir dados na coluna MyXMLDoc.

```
INSERT INTO MyXMLDocs (DocID, MyXMLDoc) Values
            (1, '<MyXMLDoc xmlns="http://MyXMLDocSchema">
                 <DocumentID>1</DocumentID>
                 <DocumentBody>"Meu Texto"</DocumentBody>
                 </MyXMLDoc>')
```

Recuperando um esquema XML registrado

O esquema armazenado pode ser listado consultando-se a view do sistema sys.xml_schema_collections, como pode ser visto na listagem que se segue:

```
SELECT * FROM sys.xml_schema_collections

SELECT XML_SCHEMA_NAMESPACE(N'dbo', N'MyXMLDocSchema') (2)
```

Nota: Ao executar o script (2) o SQL Server Management Studio Query Editor retorna um grid com um única coluna contendo um hyperlink. Clicando-se no hyperlink, exibe-se o XSD no XML editor, que mostra o resultado exibido na listagem anterior.

Suporte XQuery

O XQuery é baseado na linguagem XPath criada pela W3C (WWW.w3c.org), para consultar dados XML. XQuery estende a linguagem XPath adicionando a habilidade de atualizar dados assim como suportar uma melhor interação e classificação dos resultados. A T-SQL suporta um subconjunto da linguagem XQuery que é usada para consultar tipos de dados XML. Uma coisa boa sobre o suporte do XQuery do SQL Server é a forte integração que este possui com o mecanismo relacional. O XQuery é intimamente integrado com a T-SQL, e as consultas XML não estão restritas ao conteúdo de uma única linha XML, mas em vez disso, podemos cruzar múltiplas linhas exatamente como as consultas relacionais, sem necessidade de extrair e analisar o XML de cada linha.

Consultando elementos de Dados

O XQuery é uma flexível linguagem de consultas que é bem adaptada à consultar documentos XML que tenha uma estrutura hierárquica.

Consultando Múltiplos Elementos

O XQuery pode retornar o resultado de um nó XML ou de múltiplos nós. O exemplo a seguir ilustra o retorno de todos os subelementos e seus valores:

```
DECLARE @x xml
SET @x =
'<Myroot><Element1>One</Element1><Element2>Two</Element2></Myroot>'
SELECT @x.query('/Myroot')
```

Consultando um único Elemento

```
DECLARE @x xml
SET @x = '<Myroot><Element1>One</Element1></Myroot>'
SET @x.query('/Myroot')
```

Consultando XML Tipada

A XML tipada (isto é, XML que tenha um esquema associado) exige que você declare o namespace apropriado para recuperar os nós do documento XML. Veja o exemplo a seguir:

```
SELECT MyXMLDoc.query('declare namespace
tns="http://MyXMLDocSchema";/tns:MyXMLDoc/..') AS MyXMLBody
FROM MyXMLDocs
```

FLWR (For-Left-Where-Return)

As consultas no estilo XPath funcionam bem para consulta padrão. Entretanto, elas não são tão flexíveis quanto as T-SQL. A declaração FLWR adicionam um nível de flexibilidade à implementação XQuery do SQL Server. Você pode ver um exemplo do uso da declaração FLWR na listagem a seguir:

```
SELECT MyXMLDoc.query
('declare namespace tns="http://MyXMLDocSchema";
  for $db in /tns:MyXMLDoc
  Where /tns:MyXMLDoc/tns:DocumentID = 1
  return $db')
FROM MyXMLDocs
```

Para maiores detalhes sobre a XQuery consultar o site:

http://www.w3.org/XML/Query

http://www.w3.org/TR/2004/WD-xquery-20040723/

books OnLine do SQL Server

Métodos do tipo de dados XML

O SQL Server 2005 fornece diversos métodos internos para trabalhar com o tipo de dados XML.

Exist (XQuery)

O método de dados XQuery permite que você verifique o conteúdo de um documento XML para a existência de elementos ou atributos usando uma expressão XQuery. O método Exist recebe um parâmetro que consiste em uma declaraçãoXQuery e retorna os seguintes valores:

Valor retornado	Descrição
0	O nó não foi encontrado (false)
1	O nó existe (true)
Null	O tipo dedados XL era Null

A listagem a seguir mostra como usar o método Exist do tipo de dados XML:

```
SELECT * FROM MyXMLDocs
   WHERE MyXMLDoc.exist('declare namespace
tns="httP://MyXMLDocSchema";
     /tns:MyXMLDoc/tns:DocumentID=1') = 1
```

Modify (XML DML)

O método Modify permite que você modifique um documento XML armazenado. Você pode usar o método Modify tanto para atualizar todo um documento XML ou para atualizar apenas uma parte selecionada do documento. Veja um exemplo.

```
UPDATE MyXMLDocs SET
MyXMLDoc.modify ('declare namespace tns="httP://MyXMLDocSchema";
```

```
   replace value of (/tns:MyXMLDoc/tns:DocumentBody)[1] with
   "Modifield Body" ') WHERE DocID = 1
```

Método query (XQuery)

O método Query do tipo de dados XML pode recuperar tanto o conteúdo inteiro de um documento XML ou seções selecionadas do documento XML.

```
SELECT DocID, MyXMLDoc.query('declare namespace
tns="httP://MyXMLDocSchema";
   /tns:MyXMLDoc/tns:DocumentBody') AS Body
FROM MYXMLDocs
```

Value (XQuery, [referência do nó])

O método Value permite a extração de valores escalares de um tipo de dados XML. Veja o exemplo a seguir:

```
SELECT MyXMLDoc.value('declare namespace xd="httP://MyXMLDocSchema";
   (/xd:MyXMLDoc/xd:DocumentID)[1]', 'int') AS ID
FROM MYXMLDocs
```

Índices XML

O tipo de dados XML suporta no máximo 2GB de armazenamento, o que é bastante grande. O tamanho dos dados XML e seu uso podem ter um maior impacto na performance que o sistema pode fornecer quando consulta dados XML. Para melhorar a performance das consultas XML, o SQL Server 2005 fornece a habilidade de criar índices sobre as colunas que possuam o tipo de dados XML.

Índices primários XML

Para criar um índice primário em uma coluna do tipo de dados XML, deve existir uma chave primária agrupada (clustered) para a tabela. Você pode ter apenas um índice XML por coluna. Um índice XML não pode ter o mesmo nome de outro índice existente. Os índices XML podem ser criados apenas nos dados XML de uma tabela. O exemplo a seguir mostra como criar um índice XML na tabela MyXMLDocs:

```
CREATE PRIMARY XML INDEX MyXMLDocsInx ON MyXMLDocs MyXMLDoc)
```

Índices XML secundários

Além do índice primário, você também constrói índices XML secundários. O SQL Server 2005 suporta os seguintes índices XML secundários:

Tipo de índice secundário	Descrição
Path	O caminho do documento é usado para construir o índice
Value	Os valores do documento são usados para construir o índice
Property	As propriedades do documento são usadas para se construir o índice

A listagem a seguir mostra a criação de um índice secundário por caminho (path):

```
CREATE XML INDEX My2ndXMLDocsIdx ON MyXMLDocs (MyXMLDoc)
  USING XML INDEX MyXMLDocsIdx FOR PATH
```

Usando a Cláusula FOR XML

A cláusula FOR XML permite que o SQL Server retorne os resultados XML de uma consulta.

FOR XML RAW

O modo FOR XML RAW retorna um conjunto resultante onde cada linha do resultado é retornada em um elemento nomeado com um identificador genérico para cada linha. O valor de cada coluna é retornado usando atributos em pares. A listagem a seguir apresenta um exemplo do uso do comando FOR XML RAW:

```
USE AdventureWorks
GO
SELECT Top 3 Title, FirstName, LastName from Person.Contact FOR XML
RAW
```

FOR XML AUTO

o modo For XML Auto fornece mais flexibilidade em termos das tags que são retornadas por uma declaração XML. Entretanto, ele ainda limita a estrutura dos resultados XML que são gerados. Por padrão, usa-se o nome da Tabela ou View do SQL Server como o nome do elemento, e os nomes de colunas são usados como os

atributos para cada elemento. Você pode usar a diretiva “Elements” para especificar que cada coluna encontra-se dentro de um elemento filho.

Apesar dos resultados do modo Auto não produzirem documentos XML que se adéqüem aos padrões industriais, ele suporta sistemas de baixa adequação e podem ser usados para transferências simples B2B. A listagem a seguir apresenta um exemplo do uso do modo For XML Auto:

```
SELECT Top 3 Title, FirstName, LastName from Person.Contact FOR XML
AUTO
```

FOR XML EXPLICIT

O modo FOR XML EXPLICIT produz resultados mais flexíveis e pode ser usado para se atingir exigências mais complexas. O modo Explicit permite controle completo sobre os nomes das tags e da hierarquia e aninhamento dos elementos produzidos. As colunas podem ser mapeadas individualmente para vários elementos ou atributos. Entretanto, o modo FOR XML EXPLICIT exige o uso de consultas SQL complexas que devem especificar a estrutura de uma tabela universal que descreva o documento XML desejado. A flexibilidade do modo Explicit permite-lhe atingir as necessidades de muitas especificações-padrão industriais de mensagens.

O modo Explicit é implementado usando-se consultas UNION ALL, que essencialmente, combinam os resultados de duas ou mais consultas. Cada consulta deve conter o mesmo número de colunas e as colunas correspondentes de cada consulta precisam ter tipos compatíveis. A hierarquia XML é definida pela consulta superior (ou consulta pai). As consultas subseqüentes recuperam dados para cada um dos nós XML. A listagem a seguir mostra um exemplo do uso do modo FOR XML EXPLICIT:

```
SELECT Top 3
      1 as Tag, NULL as Parent,
      EmployeeID as [Employee!1!Employee_ID],
      NULL as [Name!2!Last_Name!ELEMENT],
      NULL as [Name!2!First_Name!ELEMENT]
FROM HumanResources.Employee E, Person.Contact C
WHERE E.ContactID = C.ContactID

UNION ALL

SELECT Top 3
     2 as Tag, 1 as Parent,
     EmployeeID,
     LastName,
     FirstName
FROM HumanResources.Employee E, Person.Contact C
WHERE E.ContactID = C.ContactID
ORDER BY [Employee!1!Employee_ID]
FOR XML EXPLICIT
```

Modo Type

Quando os tipos de dados são retornados usando-se a cláusula FOR XML do modo Type, elas são retornadas como tipos de dados XML. Veja o exemplo a seguir:

```
SELECT DocID, MYXMLDOC FROM MyXMLDocs WHERE DocID=1 FOR XML auto, type
```

Modo FOR XML PATH

O novo modo FOR XML PATH fornece um poder superior para se formatar os resultados XML tanto em relação ao modo FOR XML AUTO quanto ao FOR XML RAW, mas sem a complexidade do modo FOR XML EXPLICIT. O novo modo PATH permite aos usuários especificar o caminho na árvore XML onde um elemento ou atributo pode ser adicionado. Essencialmente, o novo modo PATH é uma alternativa mais simples para o mofo FOR XML EXPLCIT. Entretanto, ele é mais limitado que o modo EXPLICIT. Veja os exemplos a seguir:

Exemplo 1:

```
SELECT Top3 Title, FirstName, LastName FROM Person.Contact FOR XML PATH
```

Exemplo 2

```
SELECT Top 3
   Title "Employee/Title",
   FirstName "Employee/First_Name",
   LastName "Employee/Last_Name"
FROM Person.Contact FOR XML PATH
```

Consultas FOR XML aninhadas

O SQL Server 2000 era limitado ao uso da cláusula FOR XML no nível suprior de uma consulta. As subconsultas não podiam utilizar a cláusula FOR XML. O SQL Server 2005 adicionou a habilidade de se aninhar consultas FOR XML, o que é útil para se retornar múltiplos itens onde houver um relacionamento pai-filho. Um exemplo deste tipo de relacionamento pode ser os cabeçalhos de um pedido e os registros com os

detalhes do pedido; outro pode ser categorias de produtos e suas subcategorias. Veja o exemplo a seguir:

```
SELECT (SELECT Title, FirstName, LastName FROM Person.Contact
   FOR XML RAW, TYPE, ROOT('root')).query('/root[1]/row[1]')
```

Repare que a declaração SELECT usao o modo Type para retornar um resultado XML. Este resultado é então processado usando uma simples XQUERY executada com o método query do tipo de dados XML. Neste caso, a XQUERY extrai os valores da primeira linha no conjunto resultante, como é mostrado aqui:

Resultado:

```
<row Title="Mr." FirstName="Gustavo" LastName="Achong" />
```

Geração InLine do Esquema XSD

O suporte do SQL Server 2005 para a FOR XML também tem a habilidade de gerar um esquema XSD adicionando uma diretiva XMLSCHEMA para a cláusula FOR XML. Veja o exemplo a seguir:

```
USE TesteXML
GO
SELECT MyXMLDOC FROM MyXMLDocs WHERE DocID=1 FOR XML AUTO, XMLSCHEMA
```

O resultado é o a seguir:

```
<xsd:schema targetNamespace="urn:schemas-microsoft-com:sql:SqlRowSet1"
xmlns:schema="urn:schemas-microsoft-com:sql:SqlRowSet1"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:sqltypes="http://schemas.microsoft.com/sqlserver/2004/sqltypes"
elementFormDefault="qualified">
  <xsd:import
namespace="http://schemas.microsoft.com/sqlserver/2004/sqltypes"
schemaLocation="http://schemas.microsoft.com/sqlserver/2004/sqltypes/s
qltypes.xsd" />
  <xsd:import namespace="MyXMLDocSchema" />
  <xsd:element name="MyXMLDocs">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="MyXMLDOC" minOccurs="0">
          <xsd:complexType
sqltypes:xmlSchemaCollection="[TesteXML].[dbo].[MyXMLDocSchema]">
            <xsd:complexContent>
              <xsd:restriction base="sqltypes:xml">
```

```
                <xsd:sequence>
                  <xsd:any processContents="strict" minOccurs="0"
maxOccurs="unbounded" namespace="MyXMLDocSchema" />
                </xsd:sequence>
              </xsd:restriction>
            </xsd:complexContent>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Nota:

Você também pode gerar um esquema XDR (XML Data Reduce) usando a diretiva XMLDATA em combinação com a cláusula FOR XML. Entretanto, o esquema XDR foi abandonado em favor do esquema XSD.

OPENXML

Enquanto a cláusula FOR XML cria um documento XML a partir de dados relacionais, a palavra-chave OPENXML faz o inverso. A função OPENXML fornece um conjunto de linhas relacionais a partir de um documento XML. Para usar a funcionalidade de OPENXML do SQL Server, você deve primeiro chamar a stored procedure SP_xml_preparedocument, que analisa o documento XML usando o XML Document Object Model (DOM) e retorna uma handle para o OPENXML. O OPENXML fornece então uma visão em linhas do documento XML analisado. Quando você terminar de trabalhar com o documento, você chama então a stored procedure SP_xml_removedocument para liberar os recursos do sistema consumidos pelo OPENXML e o XML DOM.

O exemplo a seguir mostra como você pode usar o OPENXML em conjunto com a cláusula WITH e o novo tipo de dados XML:

```
DECLARE @hdocument int
DECLARE @doc varchar(1000)
SET @doc = '<MyXMLDoc>
              <DocumentID>1</DocumentID>
              <DocumentBody>"OPENXML Exemplo"</DocumentBody>
            </MyXMLDoc>'
EXEC sp_xml_preparedocument @hdocument OUTPUT, @doc
SELECT * FROM OPENXML (@hdocument, '/MyXMLDoc', 10)
   WITH (DocumentID varchar(4), DocumentBody varchar(50))
EXEC sp_xml_removedocument @hdocument
```

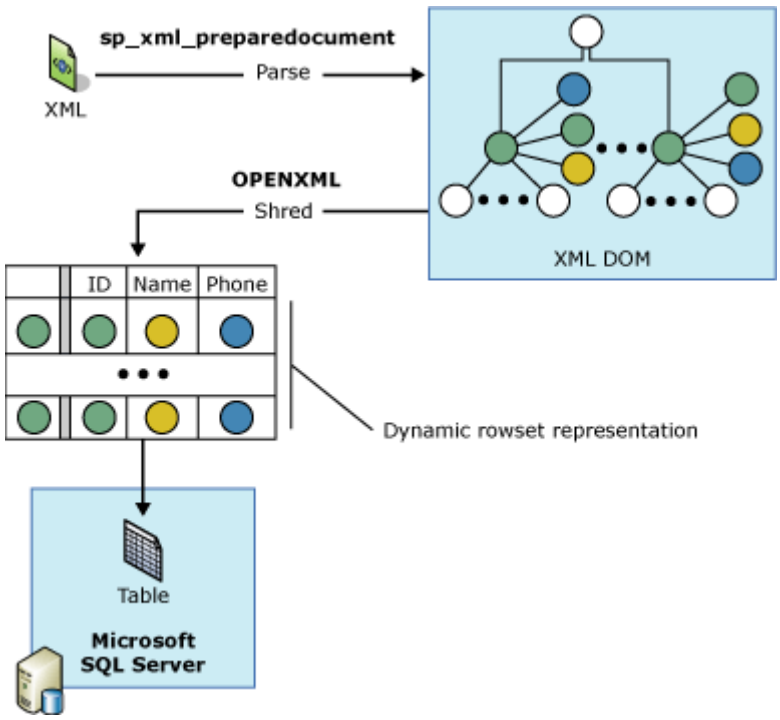

Figura 1 Representação interna de OPNXML

XML Bulk Load

Há diversas maneiras de carregar em lote (bulk load) documento XML do disco. Você pode usar o Bulk Copy Program (BCP) ou o SQL Server Integration Services. Você também pode fazer isso programaticamente usando a biblioteca de objetos SQLXML, baseada em COM, a partir do .NET ou usar a funcionalidade de carga El lote que a Microsoft adicionou à função OPENROWSET. Veja o exemplo a seguir:

```
INSERT INTO MyXMLDocs (DocId, MyXMLDoc)
  SELECT 3 as DocID, * FROM OPENROWSET (
      Bulk 'C:\temp\MyXMLDoc3.xml', SINGLE_CLOB) as DocumentID
```

Select * from MyXMLDocs //Listar o consteúdo do arquivo MyXMLDocs

Acesso nativo http SOAP

Outra nova característica encontrada no SQL Server 2005 relacionada à XML é o suporte nativo http/SOAP. Esta nova característica permite que o SQL Server responda diretamente requisições http/SOAP que são emitidas pelos Web Services sem exigir que um

sistema IIS atuem como intermediário. Usando o suporte http/SOAP, você pode criar Web Services capazes de executar lotes T-SQL, stored procedures, e funções esclares definidas pelo usuário.

Criando uma extremidade SOAP

As extremidades (endpoits) SOAP permitem o acesso programático via Web services aos objetos SQL Server como stored procedures e funções. No exemplo a seguir você verá como criar uma extremidade SOAP que expõe a stored procedure uspGetEmployeeManagers do banco de dados AdventureWorks.

```
CREATE ENDPOINT MyAdWWebService
STATE = STARTED
AS HTTP(
    PATH = '/AdWWS',
    AUTHENTICATION = (INTEGRATED),
    PORTS = ( CLEAR ),
    SITE = 'SQL2005-2'
    )
FOR SOAP (
    WEBMETHOD 'GetManagers'
      (name = 'AdventureWorks.dbo.uspGetEmployeeManagers',
       FORMAT = ROWSETS_ONLY),
    WSDL = DEFAULT,
    SCHEMA = STANDARD,
    DATABASE = 'AdventureWorks',
    NAMESPACE = 'http://AdWWS.com'
);
```

Uma vez que o endpoint tenha sido criado, ele pode ser acessado via requisição SOAP emitida por uma aplicação. Você pode listar os endpoints SOP que foram criados exibindo o conteúdo da view de sistema sys.soap_endpoints.

SELECT * FROM sys.soap_endpoints

Usando endpoints SOAP

Se o endpoint SOAP foi criado usando o valor STATE = STARTED, ele pode ser imediatamente acessado depois de o comando terminar. Entretanto, os usuários precisam ter direito de conexão ao endpoint. A sintaxe básica a declaração CONNECTION ON ENDPOINT é:

```
{ GRANT | DENY | REVOKE } CONNECTION ON ENDPOINT:: <EndPointName>
TO <login>
```

Por exemplo, o commando a seguir ilustra como você pode usar a permissão GRANT CONNECTION para permitir ao grupo Vendas se conectar a MyAdWWebService:

```
GRANT CONNECTION ON ENDPOINT:: MyAdWWebService TO HR
```

Chamando o Web Service

Após o endpoint SOAP ter sido criado e os usuários terem recebido acesso de conexão ao endpoint, você poderá chamar o Web Service a partir de sua aplicação cliente. Você pode ver a aplicação exemplo na Figura 2 .

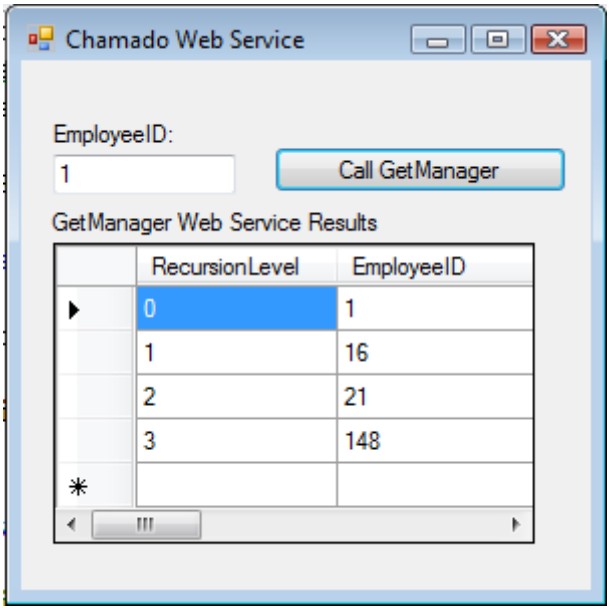

Figura 2 A aplicação cliente do Web Service

Para usar a aplicação cliente para chamar o Web service e exibir o resultado no grid, o usuário entra com o ID do funcionário (employee) e então clica o botão Call Getmanager para executar o Web service e exibir o conjunto resultante no grid.

Para criar este projeto, siga os seguintes passos:

1. Crie um novo projeto no Visual Studio “Windows Application”;
2. Arraste dois lables, um botão, TextBox, e um controle DataGridView para o Form.

3. Monte o design como o da Figura 2 acima.
4. Após arrumar os elementos da interface, você precisa adicionar uma referência ao Web service selecionando a opção Project | Add web Reference, que irá exibir uma caixa de diálogo como a mostrada na Figura 3.
5. Na URL, entre com o endereço URL mostrada na Figura3 e clique GO. Se o Visual Studio encontrar o Web Service, ele será listado na tela, como você pode ver na Fgura 3. Você pode renomear opcionalmente a referência usando a caixa de texto Web Reference Name. Então, adicione a referencia clicando no botão Add Reference.
6. Após ter adicionado a referencia ao Web service, você precisa criar o código que executará o Web service. Veja o código na listagem a seguir:

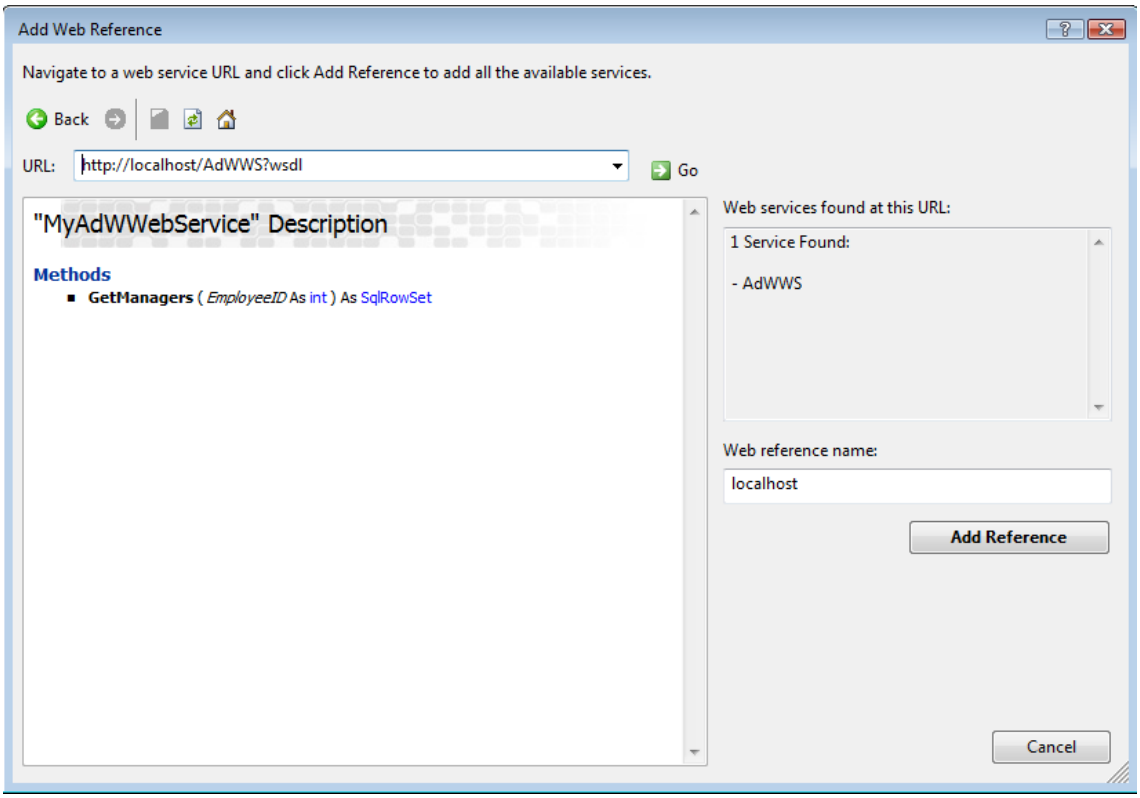

Figura 3 Adicionando uma Web referência

Listagem do botão Call Getmanager

```
//Criar uma nova instância do Web service
            localhost.MyAdWWebService MyAdWebService = new
localhost.MyAdWWebService();
            //Autenticar para usar o Web Service
            MyAdWebService.Credentials =
System.Net.CredentialCache.DefaultCredentials;

            //Converte o resulltado do Web Service para
DataSet
```

```
            DataSet ds = new DataSet();
            SqlDataAdapter da = new SqlDataAdapter();
            ds =
MyAdWebService.GetManagers(Convert.ToInt32(txtEmployeeID.Te
xt));
            //da.Fill (ds);
            dataGridView1.DataSource = ds.Tables[0];
```

Todo este laboratório foi montado como base nas informações coletadas das seguintes fontes (Bibliografia):

- Books ONLine
- Microsoft SQL Server 2005: Guia do Desenvolver. Autores: Michael Otey e Denielly Otey