Laboratorium 11 - MS SQL Server 2008

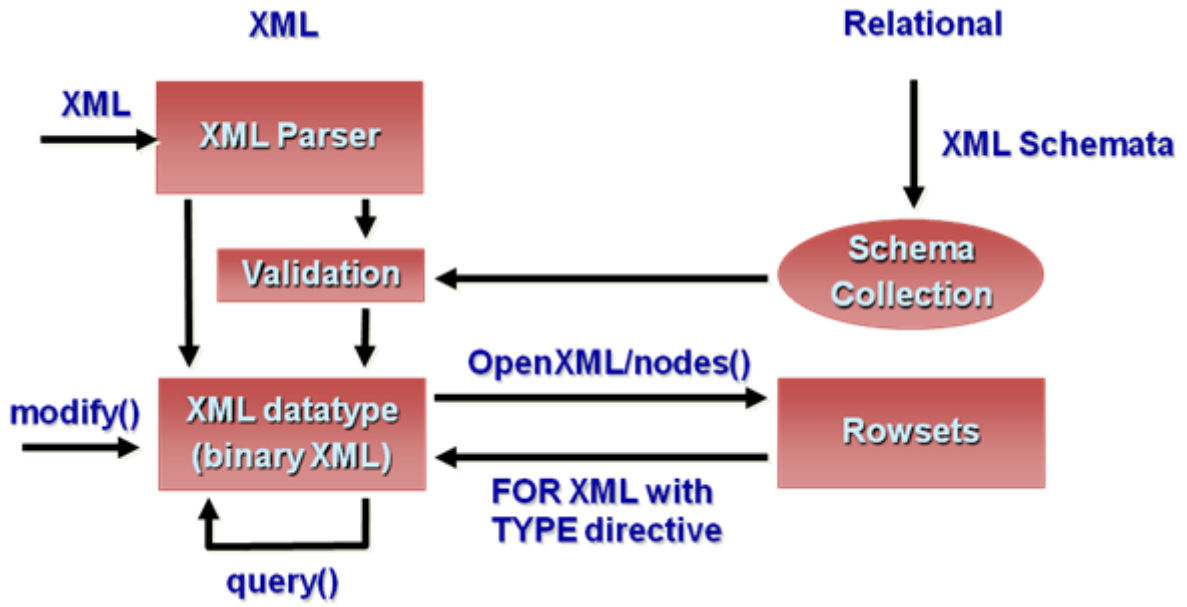
Temat: Przegląd możliwości SQL Server 2008 związanych z wsparciem dla XML
Opracowanie: A.Dydejczyk, A.Lemański

Tematyka realizowana w trakcie laboratorium:

A. Wprowadzenie
B. Typ danych XML
C. Dane XML walidowane odpowiednim schematem XML Schema
D. Publikowanie dokumentów XML z danych relacyjnych
E. Przetwarzanie dokumentów XML i danych relacyjnych

A. Wprowadzenie

Od wersji SQL Server 2005 i w następnych wersjach firma Microsoft wprowadziła nowe możliwości związane z przechowywaniem i przetwarzaniem danych XML. Ta nowa funkcjonalność ułatwia zapisywanie dokumentów w tym formacie przy zwiększonej wydajności w stosunku do wcześniej stosowanych technik. Przetwarzanie dokumentów XML realizowane jest przez wyspecjalizowane środowisko serwera oraz istnieje możliwość przetwarzania dokumentów XML i relacyjnych w oparciu o określony schemat działania (rys.1).



Rys.1 Przetwarzanie dokumentu XML w MS SQL Server

Udostępnione funkcjonalności przetwarzania dokumentów XML w bazie MS SQL Server.

1. Dedykowany typ danych o nazwie XML, którego możemy użyć do przechowania całych dokumentów i do fragmentów.
2. Możliwość zarejestrowania schematów XML w SQL i przechowania zawartości schematów w bazie danych.
3. Automatyczna walidacja dokumentu z schematem, jeżeli istnieje.
4. Automatyczny podział dokumentu XML w celu zwiększenia efektywności przeszukiwania danych XML i zmiany zawartości takiego dokumentu.
5. Implementacja podzbioru specyfikacji języka W3C XQuery i XML-DML w zakresie przeszukiwania danych XML i zmiany zawartości takiego dokumentu.
6. Możliwość tworzenia indeksów XML podstawowych (primary) i indeksów drugorzędnych ułatwiających przetwarzanie dokumentów XML.
7. Wsparcie dla .NET Common Language Runtime (CLR) w SQL Server, które umożliwia procedurom składowanym przetwarzanie dokumentów XML w kodzie zarządzalnym.

Model danych XML w MS SQL Server 2005 i 2008

Możliwości nowego typu danych ujawniają się dla danych reprezentujących, czy też raczej zawartych w kolekcjach obiektów typu **DataSet** w aplikacjach **.NET.** Usługi w takich aplikacjach dostarczają danych przenoszonych przez sieć przez protokół HTTP i mają postać strumienia tekstowego. To zapewnia, że mogą być przeszukiwane przez znane i standardowe metody i mogą dotrzeć wszędzie (HTTP z nie binarną zawartością). Jeżeli dołączy się możliwość rekonstrukcji zawartości DataSet po stronie klienta to staje się to bardzo atrakcyjne.

Do określenia typu danych w dokumencie XML służy schemat. Odbiorca na tej podstawie może określić ten typ i odpowiednio z niego skorzystać. Ten fakt jest podstawą modelu W3C **InfoSet (** link http://www.w3.org/TR/xml-infoset/).

Model ten rozważa dokumenty jako zestaw jednego lub wielu zestawów rekordów z określonymi typami danych czyli **typed rowsets.**

B. Typ danych XML

1. Typ danych XML.

```
declare @xdoc xml
set @xdoc = '<?xml version="1.0"?> <test></test>'
select @xdoc
select cast(@xdoc as varchar)
```

Sprawdzić próbę przypisania do zmiennej typu XML danych niezgodnych z regułami dokumentów XML.

2. Tworzymy tabele w bazie tempdb z kolumną XML.

```
declare @xdoc xml
set @xdoc = '<xml/>'
print cast(@xdoc as varchar)
-- Outputs: "<xml/>"
```

```
create table SampleTable (
  RecordNo  int,
  XDocument xml )
go
insert into SampleTable values (0, '<xml/>')
select * from SampleTable
```

3. Metody operowania na danych typu XML – query(), value(), exist(), modify() i nodes(). Link do dokumentacji Microsoft: http://msdn.microsoft.com/en-us/library/ms190798(v=sql.105).aspx

   a) Metoda query(wyrażenie XQuery) – zwraca część dokumentu XML – wykorzystujemy wyrażenia XQuery lub XPath.

```
declare @xdoc xml
set @xdoc = '<?xml version="1.0"?>
<test><jeden><nazwa/></jeden><dwa><nazwa/></dwa></test>'
select @xdoc.query('test/jeden')
```

   b) Metoda value(wyrażenie XQuery, typ) – pozwala na wyodrębnienie z typu danych XML wartości skalarnych (pobranie wartości o typie SQL).Parametr pierwszy XQuery lub XPath, drugi zwracany typ danych.

```
declare @xdoc xml
set @xdoc = '<?xml version="1.0"?>
<test><jeden>Adam</jeden><dwa>Marek</dwa></test>'
select @xdoc.value('(test/jeden)[1]','varchar(20)')
```

   c) Metoda exist(wyrażenie XQuery) – pozwala na sprawdzenie zawartości dokumentu XML pod kątem istnienia elementów lub atrybutów korzystających z wyrażenia XQuery.

```
declare @xdoc xml
set @xdoc = '<?xml version="1.0"?>
<test><jeden>Adam</jeden><dwa>Marek</dwa></test>'
select @xdoc.exist('test/jeden')

set @xdoc = '<root PresentationDate="2006-11-02"/>'
select @xdoc.exist(
  '/root[(@PresentationDate cast as xs:date?) eq
   xs:date("2006-11-02")]')

set @xdoc = '<PresentationDate>2006-11-02</PresentationDate>'
select @xdoc.exist(
  '/PresentationDate[(text()[1] cast as xs:date ?) =
   xs:date("2006-11-02") ]')
```

   d) Metoda nodes(wyrażenie XQuery) – stosowana do rozdzielenia danych XML na wiersze. Struktura metody: nodes (XQuery) as Table(Column)

```
declare @xdoc xml
set @xdoc = '<test><imie>Adam</imie><imie>Marek</imie></test>'
select T.c.query('.') results from @xdoc.nodes('test/imie') T(c)
```

e) Metoda modify(XMLdml) – pozwala na modyfikację przechowywanego dokumentu XML. Przy użyciu tej metody można uaktualniać cały dokument XML lub tylko jego wybraną część. Modyfikacje dokumentu XML wykonujemy z wykorzystaniem języka XML DML (Data Manipulation Language). Wyróżniamy trzy polecenia DML: insert – wstawianie węzłów XML, delete – usunięcie węzłów XML i replace value of – zastępowanie zawartości węzła inną zawartością.

```
declare @xdoc xml
-- wstawianie wezla po okreslonym wezle
set @xdoc = '<test><node>1</node><node>2</node><node>3</node></test>'
set @xdoc.modify('insert <node>4</node> after (/test/node)[3]')
select @xdoc

-- wstawianie wezla przed okreslonym wezle
set @xdoc = '<test><node>1</node><node>2</node><node>3</node></test>'
set @xdoc.modify('insert <node>0</node> before (/test/node)[1]')
select @xdoc

-- wstawianie wezla jako pierwszego w wezle
set @xdoc = '<test><node>1</node><node>2</node><node>3</node></test>'
set @xdoc.modify('insert <node>0</node> as first into (/test)[1]')
select @xdoc

-- wstawianie wezla jako ostatniego w wezle
set @xdoc = '<test><node>1</node><node>2</node><node>3</node></test>'
set @xdoc.modify('insert <node>4</node> as last into (/test)[1]')
select @xdoc

-- wstawianie wezla wewnatrz wezla
set @xdoc = '<test><node>1</node><node>2</node><node>3</node></test>'
set @xdoc.modify('insert <node1>test1</node1> into (/test/node)[1]')
select @xdoc

-- dodawanie atrybutu
set @xdoc = '<test><node>1</node><node>2</node><node>3</node></test>'
set @xdoc.modify('insert attribute val {"44"} into (/test/node)[2]')
select @xdoc

-- usuwanie wezla z dokumentu
set @xdoc = '<test><node>1</node><node>2</node><node>3</node></test>'
set @xdoc.modify('delete /test/node[2]')
select @xdoc

-- usuwanie atrybutu z wezla
```

```
set @xdoc = '<test><node
val="1">1</node><node>2</node><node>3</node></test>'
set @xdoc.modify('delete /test/node[1]/@val')
select @xdoc

-- zmiana zawartosci wezla
set @xdoc = '<test><node>1</node><node>2</node><node>3</node></test>'
set @xdoc.modify('replace value of (test/node)[2]/text()[1] with "22"')
select @xdoc

-- zmiana zawartosci atrybutu
set @xdoc = '<test><node
val="11">1</node><node>2</node><node>3</node></test>'
set @xdoc.modify('replace value of (/test/node/@val)[1] with "12"')
select @xdoc
```

Zadanie Z1

Wykorzystując metody typu XML należy utworzyć skrypt tworzący dokument XML zawierający listę studentów zawierającą imię, nazwisko i grupę. Kolejność instrukcji T-SQL tworzących dokument XML.

a) Tworzymy dokument XML zawierający element <lista>.
b) Dodajemy kolejnych studentów wykorzystując metodę „modify”. Dane studenta umieszczamy w strukturze <student><nazwisko/><imie/></student>.
c) Modyfikujemy dane studenta dodając na ostatniej pozycji w elemencie <student> element <grupa>.
d) Wykorzystując metodę „nodes” tworzymy zbiór rekordów zawierających element <student>.
e) Skrypt powinien utworzyć dokument zawierający co najmniej 5 węzłów <student/>.

C. Dane XML walidowane odpowiednim schematem XML Schema

W ramach bazy danych MS SQL Server można wprowadzać dane XML, które nie będą walidowane z wykorzystaniem XML Schema (untyped XML Data) i dane XML walidowane z zadanym schematem XML Schema (typed XML Data). W ramach niniejszego punktu omówimy zagadnienie związane z danymi XML walidowanymi odpowiednimi danymi XML Schema.

1. W ramach interfejsu graficznego możemy sprawdzić obecne w bazie danych schematy wyszukując w odpowiedniej gałęzi bazy danych. Przykładowo dostępne schematy w bazie AdventureWorks.

   Rozwijamy węzły:
   AdventureWorks -> Programability -> Types -> XML Schema Collection

   Wykorzystując polecenie T-SQL w ramach konkretnej bazy danych:
   SELECT * FROM sys.xml_schema_collections;

Przestrzenie nazw wykorzystane w ramach danych schematów wyświetlimy z wykorzystaniem następującego polecenia T-SQL.

```
SELECT * FROM sys.xml_schema_namespaces;
```

Postać schematu wyświetlana jest w jednej linii, czyli niełatwo poznać szczegóły takiego schematu. W celu przeglądania zawartości schematu w postaci bardziej czytelnej można skorzystać z poniższego polecenia.

```
SELECT xml_schema_namespace(N'relational_schema ',N'sql_identifier')
```

W ramach zapytania **'relational_schema'** jest schematem relacyjnym do którego należy XML Schema, a **'sql_identifier'** jest nazwą pod którą został on zapisany. Poniżej przykładowe polecenie dla jednego z schematów w bazie AdvantureWorks.

```
SELECT
xml_schema_namespace(N'HumanResource',N'HRResumeSchemaCollection')
```

2. Operacje tworzące i usuwające schemat można prześledzić z panelu graficznego. Wybieramy z drzewie obiektów bazy danych interesujący nas schemat a następnie wybierając prawym przyciskiem myszy menu kontekstowe i wybieramy polecenie Drop and Create to New Query Editor Windows. Otrzymamy poniższe polecenia.

```
DROP XML SCHEMA COLLECTION [relational_schema.]sql_identifier
CREATE XML SCHEMA COLLECTION [relational_schema.]sql_identifier AS'…'
```

Do modyfikacji dokumentu XML Schema wykorzystujemy polecenie – ALTER.

```
ALTER XML SCHEMA COLLECTION [ relational_schema.]sql_identifier ADD
'Schema Component'
```

[relational_schema.] - oznacza to opcjonalny element składni.

ALTER XML SCHEMA może dodawać nowy element na podstawowym poziomie, ale nie może modyfikować istniejących składowych (te są immutable). Metodą na zmianę jest zmienie**nie** kolumny XML na untyped, DROP I CREATE schematy, a następnie zmienić kolumny XML na typowane. Pociąga to za sobą także ponowne utworzenie indeksów (jeżeli były).

3. Tworzymy w bazie danych tempdb poniższy schemat ProductSchema.

```
CREATE XML SCHEMA COLLECTION ProductSchema AS '
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://www.microsoft.com/schemas/adventure-
works/products"
        xmlns:prod="http://www.microsoft.com/schemas/adventure-
works/products">
        <xs:element name="Product">
                <xs:complexType>
                        <xs:sequence>
```

```
                        <xs:element ref="prod:ProductID" />
                        <xs:element ref="prod:ProductName" />
                        <xs:element ref="prod:SupplierID" />
                        <xs:element ref="prod:CategoryID" />
                        <xs:element ref="prod:QuantityPerUnit" />
                        <xs:element ref="prod:UnitPrice" />
                        <xs:element ref="prod:UnitsInStock" />
                        <xs:element ref="prod:UnitsOnOrder" />
                        <xs:element ref="prod:ReorderLevel" />
                        <xs:element ref="prod:Discontinued" />
                    </xs:sequence>
                </xs:complexType>
            </xs:element>

            <xs:element name="ProductID" type="xs:integer" />
            <xs:element name="ProductName" type="xs:string" />
            <xs:element name="SupplierID" type="xs:integer" />
            <xs:element name="CategoryID" type="xs:integer" />
            <xs:element name="QuantityPerUnit" type="xs:string" />
            <xs:element name="UnitPrice" type="xs:double" />
            <xs:element name="UnitsInStock" type="xs:integer" />
            <xs:element name="UnitsOnOrder" type="xs:integer" />
            <xs:element name="ReorderLevel" type="xs:integer" />
            <xs:element name="Discontinued" type="xs:boolean" />
    </xs:schema>
    '
```

4. W bazie danych tempdb tworzymy tabelę dbo.ProductDocs z kolumną typem danych XML wykorzystując utworzony poprzednio schemat do walidacji danych.

```
CREATE TABLE ProductDocs (
        ID INT IDENTITY PRIMARY KEY,
        ProductDoc XML(ProductSchema) NOT NULL
)
GO
```

5. Dodajemy dane, jednocześnie sprawdzając działanie typowanej kolumny XML'a, czyli związanej ze schematem

```
INSERT INTO ProductDocs VALUES('
<Product xmlns="http://www.microsoft.com/schemas/adventure-
works/products">
        <ProductID>1</ProductID>
        <SupplierID>1</SupplierID>
        <CategoryID>1</CategoryID>
        <QuantityPerUnit>10 opak. x 20 szt.</QuantityPerUnit>
        <UnitPrice>18.0000</UnitPrice>
        <UnitsInStock>39</UnitsInStock>
        <UnitsOnOrder>0</UnitsOnOrder>
```

```
        <ReorderLevel>10</ReorderLevel>
        <Discontinued>0</Discontinued>
</Product>
')
```

Wprowadzając powyższy rekord otrzymamy komunikat z SQL Server'a podobny do poniższego. Powodem błędu jest brak elementu ProductName

```
--Msg 6965, Level 16, State 1, Line 1
--XML Validation: Invalid content.
--Expected element(s):http://www.microsoft.com/schemas/adventure-
works/products:ProductName
--where element 'http://www.microsoft.com/schemas/adventure-
works/products:SupplierID' was specified.
--Location: /*:Product[1]/*:SupplierID[1]
```

6. Następnie podajemy już poprawne dane

```
INSERT INTO ProductDocs VALUES('
<Product xmlns="http://www.microsoft.com/schemas/adventure-
works/products">
        <ProductID>1</ProductID>
        <ProductName>Herbata</ProductName>
        <SupplierID>1</SupplierID>
        <CategoryID>1</CategoryID>
        <QuantityPerUnit>10 opak. x 20 szt.</QuantityPerUnit>
        <UnitPrice>18.0000</UnitPrice>
        <UnitsInStock>39</UnitsInStock>
        <UnitsOnOrder>0</UnitsOnOrder>
        <ReorderLevel>10</ReorderLevel>
        <Discontinued>0</Discontinued>
</Product>
')
```

7. Sprawdzenie poprawności wprowadzonych danych.

```
SELECT * FROM ProductDocs
SELECT * FROM ProductDocs FOR XML AUTO
SELECT * FROM ProductDocs FOR XML RAW
```

Zadanie Z2

W ramach zadania należy opracować skrypt T-SQL tworzący tabelę zawierającą następujące atrybuty: id integer PK, nazwisko varchar(30), imie varchar(20), adres XML. Typ dokument XML powinien zawierać element <adres> oraz elementy potomne <miejscowość>, <kod>, <ulica> oraz <numer_domu> i <numer_mieszkania>. Należy przygotować XML Schema do dokumentu XML (numer_mieszkania może być opcjonalny). W ramach skryptu należy dodać przykładowy rekord danych i sprawdzający poprawność wstawienia danych do tabeli.

D. Publikowanie dokumentów XML z danych relacyjnych

Tworzenie dokumentów XML w ramach bazy danych MS SQL Server wykonywane jest z wykorzystaniem polecenia SELECT .. FROM … FOR XML … . Poniżej przedstawione zostaną przykłady wykorzystania polecenia do tworzenia dokumentów XML w bazie danych AdventureWorks.

W ramach ćwiczenia zapoznamy się z czterema formami polecenia FOR XML.

a) FOR XML RAW
b) FOR XML AUTO
c) FOR XML EXPLICIT
d) FOR XML PATH

1. Tworzenie dokumentu XML przy pomocy polecenia FOR XML RAW
   Każdy wiersz zwracanego wyniku zamieniany jest na jeden element XML, a wartości poszczególnych kolumn na atrybuty

```
SELECT * FROM sales.Customer FOR XML RAW
```

2. Tworzenie dokumentu XML przy pomocy polecenia FOR XML AUTO, ROOT
   Klauzla XML AUTO działa podobnie jak XML RAW, jednak gdy zwracany wynik jest rezultatem złączenia wielu tabel, pokazuje on strukturę danych. Klauzula ROOT(„nazwa elementu”) – tworzy element główny. Klauzula ta nie może być użyta z klauzulą XMLDATA i XMLSCHEMA

```
SELECT * FROM sales.Customer FOR XML AUTO, ROOT('customers')
```

3. Tworzenie dokumentu XML przy pomocy polecenia FOR XML AUTO, ELEMENTS XSINIL Klauzula ELEMENTS w trybach AUTO i RAW powoduję, że wartości kolumn zostaną zamienione na elementy zamiast na atrybuty. Klauzula XSNILL – domyślnie pusty element jest pomijany. W przypadku wykorzystania z klauzuli ELEMENTS I XSNILL dołączony zostanie element pusty z atrybutem xsi:nill=„1”.

```
SELECT * FROM sales.Customer
      FOR XML AUTO, ELEMENTS XSINIL, ROOT('customers')
```

4. Tworzenie odpowiedzi XML z wykorzystaniem konstrukcji FOR XML PATH()
   Format tworzonego dokumentu XML jest określony przez wyrażenia XPath.

```
SELECT EmployeeID "@EmpID",
            FirstName "EmpName/First",
            MiddleName "EmpName/Middle",
            LastName "EmpName/Last"
      FROM   HumanResources.Employee E, Person.Contact C
      WHERE E.EmployeeID = C.ContactID AND E.EmployeeID=218
      FOR XML PATH
SELECT SalesOrderID as "data()"
       FROM Sales.SalesOrderHeader
       WHERE SalesOrderHeader.ContactID = Contact.ContactID
```

```
        FOR XML PATH (''))

SELECT ContactID as "@ContactID",
        FirstName as "@ContactName",
        (SELECT SalesOrderID as "data()"
            FROM Sales.SalesOrderHeader
            WHERE SalesOrderHeader.ContactID = Contact.ContactID
            FOR XML PATH ('')) as "@SalesOrderIDs"
    FROM Person.Contact
    FOR XML PATH('SalesOrders')
```

5. Tworzenie zapytania z konstrukcją FOR XML EXPLICIT. W zapytaniu strukturę zwracanego wyniku należy zdefiniować z wykorzystaniem dodatkowych informacji. W ramach zwracanego dokumentu XML można tworzyć elementy, atrybutu i sekcje CDATA.

```
SELECT
   1 AS Tag,
   0 AS Parent,
   ContactID        as [Contact!1!ContactID],
   FirstName        as [Contact!1!FirstName!ELEMENT],
   MiddleName       as [Contact!1!MiddleName!ELEMENTXSINIL],
   LastName         as [Contact!1!LastName!ELEMENT],
   '<ContactInformation>Contact information</ContactInformation>'
                                    as [Contact!1!!cdata]
FROM   Person.Contact
WHERE   ContactID=483
FOR XML EXPLICIT
```

6. Tworzenia zapytania z konstrukcja FOR XML AUTO, TYPE
   Rezultat zapytania będzie zmienną typu XML, zamiast zmienną tekstową. Pozwala to na pisanie złożonych zapytań przy użyciu trybów RAW i AUTO.

```
SELECT HumanResources.Employee.Title,
        HumanResources.Employee.Birthdate
    FROM HumanResources.Employee
    WHERE Employee.ContactID = Contact.ContactID
    FOR XML AUTO, TYPE, ELEMENTS

SELECT ContactID, FirstName, LastName,
        (SELECT HumanResources.Employee.Title,
                HumanResources.Employee.Birthdate
            FROM HumanResources.Employee
            WHERE Employee.ContactID = Contact.ContactID
            FOR XML AUTO, TYPE, ELEMENTS)
    FROM Person.Contact
    WHERE ContactID > 1000
    ORDER BY ContactID
FOR XML AUTO, TYPE
```

7. Tworzenie XML Schema w zapytaniu SELECT.
   Do wynikowego XML'a zostanie dołączony schemat XSD opisujące dane.

```
SELECT ContactID, FirstName, LastName
      FROM Person.Contact
      WHERE ContactID = 218
     FOR XML AUTO, XMLSCHEMA

SELECT Contact.ContactID,
           SalesOrderHeader.SalesOrderID,
           SalesOrderHeader.ContactID,
           Contact.FirstName,
           Contact.MiddleName
     FROM Sales.SalesOrderHeader, Person.Contact
     WHERE SalesOrderHeader.ContactID = Contact.ContactID
              AND  Contact.ContactID = 226
     FOR XML RAW, XMLSCHEMA, ELEMENTS XSINIL
```

   Więcej przykładów można znaleźć pod adresem:
   https://www.simple-talk.com/sql/learn-sql-server/using-the-for-xml-clause-to-return-query-results-as-xml/

E. Przetwarzanie dokumentów XML i danych relacyjnych

W ramach tego ćwiczenia wykorzystamy metody typu danych XML nie tylko do danych zawartych w polach XML bazy danych ale także utworzonych w poleceniu SQL przy pomocy klauzuli FOR XML. Ćwiczenie realizujemy w bazie danych AdvantureWorks.

1. Odpytywanie tabeli z polem XML

```
SELECT * FROM Sales.Individual WHERE CustomerID = 11000
```

2. Odpytywanie pola XML z wykorzystaniem metody query i polecenia XPATH

```
SELECT TOP 10 Demographics.query('
      declare default element namespace
      "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
      works/IndividualSurvey";
      /IndividualSurvey/YearlyIncome')
FROM Sales.Individual

SELECT TOP 10 Demographics.value('
      declare default element namespace
      "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
      works/IndividualSurvey";
      (/IndividualSurvey/YearlyIncome)[1]',
      'varchar(250)')
FROM Sales.Individual
```

3. Sprawdzamy zawartość tabel Person.Address i HumanResources.Employee w bazie danych AdvantureWorks.

```
SELECT * FROM  Person.Address
SELECT * FROM  HumanResources.Employee
```

4. Tworzymy zmienną XML, umieszczamy w niej wynik działania polecenia SELECT z wyrażeniem FOR XML.

```
DECLARE @x XML
SET @x = (SELECT * FROM Person.Address WHERE AddressID = 1
            FOR XML AUTO,  ELEMENTS, TYPE)
SELECT @x
```

5. Wyszukanie określonych danych w dokumencie XML metoda query().

```
DECLARE @x XML
SET @x = (SELECT TOP 50 City FROM Person.Address
            FOR XML AUTO, ELEMENTS, TYPE)
SELECT @x.query('
    <Cities>  {
        for $city in /Person.Address/City
        return $city
    }  </Cities>
')
```

6. Wykorzystanie zagnieżdżonych poleceń SELECT do odpytywania bazy danych.

```
DECLARE @x XML
SET @x = (
    SELECT Employee.*, EmployeeAddress.AddressID, (
        SELECT * FROM Person.Address Address
        WHERE EmployeeAddress.AddressID = Address.AddressID
        FOR XML  AUTO, ELEMENTS, TYPE
    ) AS Addresses
   FROM HumanResources.Employee AS Employee
   INNER JOIN HumanResources.EmployeeAddress AS EmployeeAddress
   ON Employee.EmployeeID = EmployeeAddress.EmployeeID
   WHERE Employee.EmployeeID = 1
    FOR XML AUTO, ELEMENTS, TYPE
)
SELECT @x
```

Zadanie Z3

Opracować zapytanie zwracające rekordy zawierające wartość elementu TotalChildren większe od 1 znajdującego się w dokumencie XML w atrybucie Demographics typu XML w tabeli Sales.Indyvidual.