Tallinna Ülikool
Informaatika Instituut

XML’i kasutamine SQL Server 2005 Express Edition’is

Seminaritöö

Autor: Roman Inostrantsev
Juhendaja: Jaagup Kippar

Autor…….....……...”…….”……2009. a.
Juhendaja……...”…….”……2009. a.

Tallinn 2009

Вступление

Эта работа об использовании XML в SQL Server 2005 Express Edition. В отличии от многочисленных туториалов и статей в интернете, я старался подробно объяснить порядок работы функций и комманд сервера, связанных с языком разметки. В работе не объясняется принцип работы програмного обеспечения, и устройство XML и SQL Server'а. Этой информации много в интернете, хотя многие ресурсы мне показались совсем бесполезными или слишком сложными, в которых показан только конечный результат. Для меня, как новичка, было важно знать значение каждого параметра, элемента и команды. Поэтому моя работа является обучающим материалом для решивших познать данную тему.

Данная тема очень актуальна и важна для предприятий, где хранится и обрабатывается огромное количество информации. XML упрощает хранение и передачу данных как внутри предприятий так и между удаленными клентами.

Моей задачей было подробно узнать и объяснить:

- создание схемы коллекций
- создание таблиц и их изменение
- просмотр таблиц, а именно перевод реляционных данных из таблицы в XML формат с помощью FOR XML,
- наоборот вывод данных из документа в реляционную таблицу, т.е. OPENXML
- методы типа данных, их использование в управлении данными

Тема очень обширная, я выбрал лишь самые азы, самые нужные и важные элементы управления XML данными в сервере.

1. Введение

XML - язык разметки, предназначенный для хранения данных, транспортировки данных между несовместимыми системами, широко распространенный стандарт для обмена данными между разными приложениями по локальным сетям и Internet. А также стал распространенным форматом для хранения и передачи информации в базах данных. Встроенная поддержка типа данных XML и XQuery помогают организациям удобно соединить внутренние и внешние системы. SQL Server 2005 имеет встроенную поддержку реляционных и XML данных.

Экземплярами типа данных XML могут быть столбцы таблицы, аргументы функций и хранимых процедур, а также их переменные. Кроме того, можно сделать тип данных XML специализированным, указав XML-схему, которая обеспечивает ограничения при проверке и информацию о типе данных для экземпляра XML. XML позволяет моделировать сложные данные вместо того, чтобы ограничиваться скалярными значениями, которые поддерживает SQL Server. По этой причине встроенные строковые типы данных, char или varchar, не позволяют полной и эффективной функциональности и многих преимуществ XML. При хранении в строке можно выбрать и вставить целый документ, или получить некоторые его части, однако нельзя делать запросы по содержимому документа. SQL Server 2005 позволяет делать запросы к части XML документа, проверять, что документ соответствует схеме XML, и даже изменять содержимое документа XML прямо в столбце.

После получения XML данных существует три способа их хранения:

1. хранить XML в том виде как он есть
2. перевести XML и хранить в виде реляционных данных, используя функцию OpenXML
3. хранить XML в виде текста.

SQL Server 2005 поддерживает следущие типы данных: char, varchar, text, integer, nvarchar, nchar, ntext, xml. XML является встроенным типом данных.
В работе используются:

- MS SQL Server 2005 Express Edition - бесплатная версия сервера, которая ограничена одним процессором, одним GB для буферной памяти и размером баз данных в 4GB.
- MS SQL Server Management Studio Express – бесплатное програмное обеспечение для работы с сервером.

Версия Express Edition предназначена для учеников и начинающих программистов, поэтому в ней отсуствуют некоторые инструменты. Информация и документация по SQL Server: http://www.microsoft.com/Sqlserver/2005/en/us/express.aspx
Информация, требовании и загрузка: http://www.microsoft.com/downloads/details.aspx?FamilyId=C243A5AE-4BD1-4E3D-94B8-5A0F62BF7796&displaylang=en

2. Тип данных XML

Теперь XML является встроенным типом данных для SQL Server 2005. Он имеет набор методов, позволяющих использовать возможности XQuery для запроса, проверки и правки данных.

С XML может быть связан набор XML схем (XML Schema Collection), что обеспечивает проверку ограничений, вставки, обновления и типизацию значений, хранимых внутри XML данных. Если данные типа XML связаны с набором схем XML, они называются типизированными данными XML (typed XML data), иначе они называются нетипизированными данными XML (untyped XML data).
Пример создания типизированного XML:

```
CREATE TABLE nimi (Col XML not null)
```

“Col” - столбец с нетипизированным XML.
Пример создания нетипизированного XML:

```
CREATE TABLE nimi (
        ID INT PRIMARY KEY,
        Document XML(CONTENT myCollection))
```

Создается таблица “nimi” со столбцом типизированного XML Document, используя схему myCollection.

3. XML Schema Collection

SQL Server 2005 позволяет связать схему с конкретным столбцом XML. Схема проверяет, верно ли данные XML вставлены в поле. В SQL Server 2005 поддерживается возможность группирования схем в коллекцию; это позволяет применять к разные схемы. XML сохраняется в поле XML, пройдя успешно проверку любой схемой.

Столбцы, переменные и параметры XML могут быть типизированы в соответствии с набором схем XML, которые могут быть связаны (например, используя <xs:import>) или не связаны друг с другом. Каждый типизированный экземпляр XML устанавливает пространство имен из набора схем XML, относящихся к нему. Движок базы данных проверяет экземпляр в соответствии со схемой XML во время изменения данных.

Информация схемы XML используется при хранении и оптимизации запросов. Типизированные экземпляры XML содержат типизированные значения во внутреннем, двоичном представлении, как и в индексах XML. Это обеспечивает эффективную обработку типизированных данных XML.[1]

3.1 Работа со схемами

Сначала создадим таблицу с названием "uni":

```
create table uni(osaknimi varchar(20), yli varchar(3), tnimi varchar(20), tperenimi varchar (20), Col xml(dbo.University))
```

, которая будет содержать в себе имя "Col", это новый тип данных XML. Я обозначил схему в колонку "Col", чтобы при связи с XML, проверялась совместимость со схемой.

[1] http://www.sql.ru/subscribe/2004/226.shtml . 27.02.2009

ROMA\SQLEXPRE...ter - dbo.uni | Summary

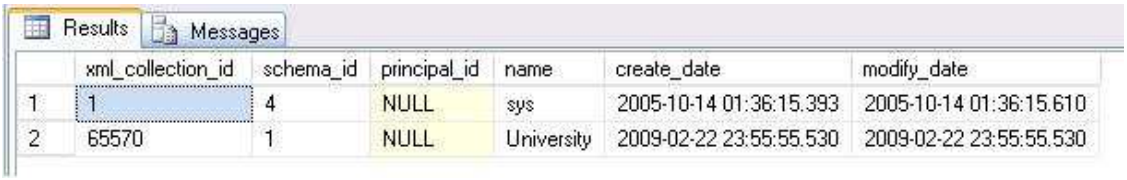
osaknimi	yli	tnimi	tperenimi
informaatika	tly	margret	hunt
informaatika	tly	arnold	tuld
fyysika	ty	lilian	must
informaatika	tly	helen	porgand
matemaatika	tty	liis	aed
fyysika	ty	martin	mai

Чтобы связать схему "University" с колонкой таблицы, надо сначала добавить схему в SQL Server 2005. Коллекция схем XML создается с помощью CREATE XML SCHEMA COLLECTION и содержит одну или более схем XML:

```
CREATE XML SCHEMA COLLECTION University AS N'
      <xsd:schema
            attributeFormDefault="unqualified"
            elementFormDefault="qualified"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:element name="Ylikool">
      <xsd:complexType>
       <xsd:sequence>
        <xsd:element name="Osakond">
         <xsd:complexType>
            <xsd:sequence>
             <xsd:element name="Nimi" type="xsd:string"/>
             <xsd:element name="Perenimi" type="xsd:string"/>
            </xsd:sequence>
         </xsd:complexType>
        </xsd:element>
       </xsd:sequence>
      </xsd:complexType>
      </xsd:element>
      </xsd:schema>'
```

Увидеть созданные схемы помогает запрос sys.XML_schema_collections:

```
select * from sys.xml_schema_collections
```



Results | Messages

	xml_collection_id	schema_id	principal_id	name	create_date	modify_date
1	1	4	NULL	sys	2005-10-14 01:36:15.393	2005-10-14 01:36:15.610
2	65570	1	NULL	University	2009-02-22 23:55:55.530	2009-02-22 23:55:55.530

Изменяется схема с помощью:

```
ALTER XML SCHEMA COLLECTION University ADD ‘компонент схемы’
```

Удаляется схема с помощью:

```
DROP XML SCHEMA COLLECTION University
```

4. XML и таблица

4.1 FOR XML clause

Для начала заполним нашу таблицу данными, для этого пользуемся командой:

```
insert into uni (osaknimi, yli, tnimi, tperenimi)
values ('fyysika','ty','lilian','tuld')
/*osaknimi - название факультета, yli - университет(ty-TÜ, tly - TLÜ, tty - TTÜ), tnimi - имя студента, tperenimi - фамилия студента*/
```

Просмотр таблицы в виде XML осуществляется с помощью команды FOR XML. Существует три способа FOR XML:

- RAW
- AUTO
- EXPLICIT

RAW представляет запрос в виде <row> элемента, где нет иерархии, а данные таблицы представлены в виде аттрибута. Запрос:

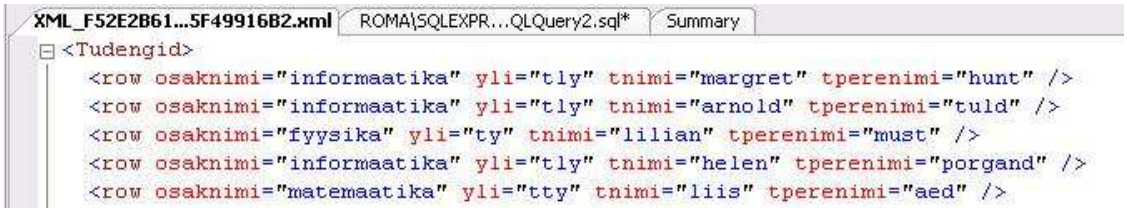
```
select osaknimi, yli, tnimi, tperenimi
from uni
for xml raw
```

выводит результат:

Путем определения параметра ROOT в запросе FOR XML, задается имя корневого элемента:

```
select osaknimi, yli, tnimi, tperenimi
from uni
for xml raw, root ('Tudengid')
```

Результат:



Если в конце запроса добавить параметр ELEMENTS,

```
select osaknimi, yli, tnimi, tperenimi
from uni
for xml raw, elements
```

то созданные элементы разбиваются на отдельные элементы:

AUTO представляет таблицу элементом, а каждый столбец этой таблицы является аттрибутом этого элемента, который называется именем таблицы.

```
select osaknimi, yli, tnimi, tperenimi
from uni
for xml auto
```

Результат:

```
<uni osaknimi="informaatika" yli="tly" tnimi="margret" tperenimi="hunt" />
<uni osaknimi="informaatika" yli="tly" tnimi="arnold" tperenimi="tuld" />
<uni osaknimi="fyysika" yli="ty" tnimi="lilian" tperenimi="must" />
```

Так же как и в случае с RAW, путем добавления параметра ROOT в конце запроса, определяется корневой элемент, а параметр ELEMENTS создает из аттрибутов отдельные элементы.

EXPLICIT позволяет контролировать форматирование XML, путем задания дополнительных сведений. Для этого создается запрос SELECT, который имеет определенный формат: создаются два столбца метаданых: Tag, Parent. Они определяют иерархию документа. Tag предоставляет номер текущего элемента. Parent задает номер для родительского элемента.

После двух обязательных полей tag и parent следуют поля, которые требуется выбрать из таблицы. Для них должен быть задан псевдоним, определяющий тип XML-узла, его название и другую информацию. Вот синтаксис этого псевдонима: `ElementName!TagNumber!AttributeName!Directive`:

- ElementName – элемента, в котором находится элемент текущего поля. Обычно здесь указывается имя таблицы.
- TagNumber - идентификатор таблицы, в которой находится данное поле.
- AttributeName – имя атрибута (или элемента), представляющего текущее поле.
- Directive – по существу, представляет собой тип узла. Может принимать следующие значения:
    1) element - поле представляется в виде элемента
    2) xml - тоже самое что и element, но не выполняет трансформацию текста
    3) cdata
    4) hide - скрывает поле

Если не указывать directive, то данные столбцов будут представлены в виде аттрибутов главного элемента. EXPLICIT выводит XML документ с данными из нескольких таблиц, путем связывания полей столбцов по id. Для примера я не буду использовать существующую таблицу, а создам 3 новых: tudeng(id, tudengi_perenimi), num(id, matrikel), hinded(id, keskhinne). Данный запрос соединяет таблицы в одну:

```
SELECT 1 AS Tag, null AS Parent,
      tudeng.id AS [Tudeng!1!id],
      tudengi_perenimi AS [Tudeng!1!perekonnanimi],
      matrikel AS [Tudeng!1!Matrikel!element],
      keskhinne as [Tudeng!1!Keskhinne!element]
FROM tudeng join num on tudeng.id=num.id
      join hinded on num.id=hinded.id
order by 3,2
```

Устанавливается tag = 1, parent = null, означая,что элементы выбранные из таблицы под первым tag не имеют родителя. Дальше даются псевдонимы элементам и

аттрибутам. Эти данные берутся из соединенных таблиц, которые связаны между собой по их id. Из этого SELECT'а получается такая таблица:

	Tag	Parent	Tudeng!1!id	Tudeng!1!perekonnanimi	Tudeng!1!Matrikel!element	Tudeng!1!Keskhinne!element
1	1	NULL	1	toomingas	98283	3.58
2	1	NULL	2	meri	45678	1.34
3	1	NULL	3	porgand	10239	4.99

При добавлении в конец запроса строки `FOR XML EXPLICIT`, получается XML:

```
<Tudeng id="1" perekonnanimi="toomingas">
  <Matrikel>98283</Matrikel>
  <Keskhinne>3.58</Keskhinne>
</Tudeng>
<Tudeng id="2" perekonnanimi="meri">
  <Matrikel>45678</Matrikel>
  <Keskhinne>1.34</Keskhinne>
</Tudeng>
<Tudeng id="3" perekonnanimi="porgand">
  <Matrikel>10239</Matrikel>
  <Keskhinne>4.99</Keskhinne>
</Tudeng>
```

4.2 OPENXML clause

OPENXML позволяет представить XML документ в виде реляционной таблицы.

Синтаксис:

```
OPENXML(idoc int[in], rowpattern nvarchar[in],[flags byte[in]])
      [WITH(SchemaDeclaration|TableName)]
```

- idoc – дескриптор XML документа, полученный при помощи процедуры sp_xml_preparedocument
- rowpattern – XPath выражение
- flags – набор флагов, определяющих каким образом сопоставляются данные XML документа и реляционного набора строк
- SchemaDeclaration – определение полей реляционного набора строк в формате:
  - `ColName ColType [ColPattern]`

Где

`ColName` – имя поля

`ColType` – тип поля. Допускаются все типы SQL Server

`ColPattern` - локализуемая группа XPath для поля.

“Считывает входной XML-текст, проводит его синтаксический анализ при помощи синтаксического анализатора MSXML (Msxmlsql.dll) и выдает проанализированный документ, готовый к потреблению. Проанализированный документ является древовидным представлением различных узлов в XML-документе: элементов, атрибутов, текста, комментариев и т. д.

Процедура **sp_xml_preparedocument** возвращает дескриптор, по которому можно обратиться к вновь созданному внутреннему представлению XML-документа. Дескриптор действует только в течение сеанса либо до тех пор, пока не будет аннулирован путем выполнения процедуры **sp_xml_removedocument**.”[2]

Синтаксис:

```
Sp_xml_preparedocument @iDoc OUTPUT, @xml_text
```

Где iDoc - дескриптор созданного документа, xml_text - исходный XML документ.

[2] http://msdn.microsoft.com/ru-ru/library/ms187367.aspx , 23.03.2009

Флаг может принимать значения 0, 1, 2. 0 и 1 – attribute-centric mapping - по умолчанию имена колонок получаемой реляционной таблицы соответствуют аттрибутам в XML документе. 2 – element-centric mapping - имена колонок таблицы будут соответствовать именам вложенных XML-элементов.

Это пример с attribute-centric mapping (flag=1). Вставленный XML получен функцией FOR XML RAW:

```
DECLARE @xml_text VARCHAR(4000), @i INT
SELECT @xml_text =
'<root>
      <row osaknimi="informaatika" yli="tly" tnimi="margret"
            tperenimi="hunt" />
      <row osaknimi="informaatika" yli="tly" tnimi="arnold"
            tperenimi="tuld" />
      <row osaknimi="fyysika" yli="ty" tnimi="lilian"
            tperenimi="must" />
      <row osaknimi="informaatika" yli="tly" tnimi="helen"
            tperenimi="porgand" />
      <row osaknimi="matemaatika" yli="tty" tnimi="liis"
            tperenimi="aed" />
      <row osaknimi="fyysika" yli="ty" tnimi="martin"
            tperenimi="mai" />
      <row osaknimi="informaatika" yli="tly" tnimi="mari"
            tperenimi="laps" />
      <row osaknimi="informaatika" yli="tly" tnimi="ants"
            tperenimi="kokk" />
      <row osaknimi="matemaatika" yli="tty" tnimi="viljar"
            tperenimi="pomm" />
      <row osaknimi="fyysika" yli="ty" tnimi="kristjan"
            tperenimi="komm" />
</root>'

EXEC sp_xml_preparedocument @i OUTPUT, @xml_text
SELECT
      osaknimi as osakond,
      yli as ylikool,
      tnimi as tudengi_nimi,
      tperenimi as tudengi_perekonnanimi
```

```
/*для лучшего восприятия я переименовал названия столбцов по своему
усмотрению*/
FROM
    OPENXML(@i, '/root/row',1) WITH (
            osaknimi varchar(20),
            yli varchar(3),
            tnimi varchar(20),
            tperenimi varchar(20)
      )

EXEC sp_xml_removedocument @i
```

Результат:

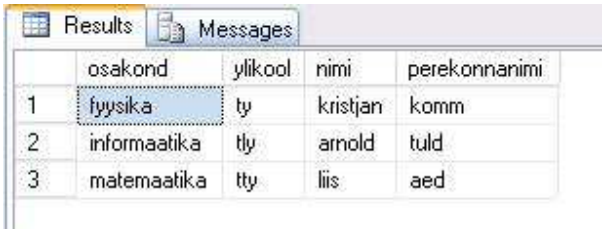
	osakond	ylikool	tudengi_nimi	tudengi_perekonnanimi
1	informaatika	tly	margret	hunt
2	informaatika	tly	arnold	tuld
3	fyysika	ty	lilian	must
4	informaatika	tly	helen	porgand
5	matemaatika	tty	liis	aed
6	fyysika	ty	martin	mai

Это пример с element-centric mapping (flag=2). Вставленный XML получен функцией FOR XML EXPLICIT:

```
DECLARE @xml_text VARCHAR(max), @i INT
SELECT @xml_text =
'<root>
 <osaknimi osaknimi="fyysika">
  <yli>
    <yli>ty</yli>
    <tnimi>kristjan</tnimi>
    <tperenimi>komm</tperenimi>
  </yli>
</osaknimi>
<osaknimi osaknimi="informaatika">
  <yli>
    <yli>tly</yli>
    <tnimi>arnold</tnimi>
```

```
    <tperenimi>tuld</tperenimi>
  </yli>
</osaknimi>
<osaknimi osaknimi="matemaatika">
  <yli>
    <yli>tty</yli>
    <tnimi>liis</tnimi>
    <tperenimi>aed</tperenimi>
  </yli>
</osaknimi>
</root>'
EXEC sp_xml_preparedocument @i OUTPUT, @xml_text
SELECT
osaknimi as osakond,
yli as ylikool,
tnimi as nimi,
tperenimi as perekonnanimi
FROM
    OPENXML(@i, '/root/osaknimi/yli',2) WITH
(osaknimi varchar(20) '../@osaknimi',
yli varchar(3),
tnimi varchar(20),
tperenimi varchar(20))
EXEC sp_xml_removedocument @i
```

Результат:



	osakond	ylikool	nimi	perekonnanimi
1	fyysika	ty	kristjan	komm
2	informaatika	tly	arnold	tuld
3	matemaatika	tty	liis	aed

5. Методы типа данных XML

Существует 5 методов:

- .query - доступ к XQuery-форматированному запросу;
- .exist - проверка данных на наличие путем вывода значения Boolean;
- .value - доступ к значению в пределах определенного элемента или аттрибута;
- .modify - изменение данных;
- .nodes - разбивает данные XML в индивидуальные ряды.

5.1. .query()

XQuery - сокращено от XML Query - язык запросов для обработки данных в форме XML, главная роль которого доставать информацию из баз данных. XQuery занимается не только XML файлами, но и всем, что может походить на XML документ, в том числе и базами данных.

“XQuery используется для:

- извлечения информации для использования в веб-сервисах;
- трансформировать XML в XHTML;
- генерировать суммарные рапорты.

XQuery использует:

- Path выражения
- конструкторы элементов
- FLWOR (“flower”) выражения
- выражения списка
- условные выражения
- количественно-определенные выражения
- выражения типа данных”[3]

ОператорыXQuery :

[3] http://www.w3schools.com/xquery/xquery_example.asp , 21.02.2009

- арифметические операторы
- операторы сравнения
- логические операторы

Тип	**Оператор**
арифметические операторы	+,-,*,div, mod
основные операторы сравнения	=, !=, <, >, <=, >=
оператор сравнения ценностей	eq, ne, lt, gt, le, ge
оператор сравнения узла	Is
оператор сравнения порядка узлов	>>, <<
логические операторы	and, or

5.1.1. Path выражения

XPath - язык нахождения информации по документу XML, использует выражения пути для выбора узлов. Существует 7 видов узлов. Главный, т.е. <root> элемент называется узлом документа, “document node”. Дальше следуют: element, attribute, text, namespace, processing-instruction и comment.

Так как выражений пути больше ста, то я не буду все перечислять и объяснять, но покажу лишь главные, которые собираюсь использовать.

Выражение	**Описание**
название_узла	Выбирает все дочерние узлы прописанного узла
/	Выбирает из главного узла
//	Выбирает узлы в документе из текущего узла, независимо от их нахождения.
.	Выбирает текущий узел
..	Выбирает родительский узел текущего узла

@	Выбирает аттрибут элемента
Osakonna/yli[1]	выбирает первый элемент из всех имеющихся с подобным именем, которые являются дочерними элементами элемента <Osakonna>
Атомарные значения	пустые узлы, элементы и аттрибуты Например: <tudengi_nimi/>
Perekonnanimi/text()	выбирает все текстовые узлы текущего узла

5.1.2. FLWOR выражения

Позволяет создавать запрос в похожем на синтаксис SQL виде. XQuery приравнивается к выражению SELECT и называется FLWOR, по буквам его параметров: for, let, where, order by, return.

- For - выполняется итерация над последовательностью введенных значений, по очереди связывая переменную с каждым значением
- let: задается переменная и присваивается ей значение, которое может быть списком, содержащим несколько элементов
- where: определяются критерии фильтрации результатов запроса
- order by: определяется порядок сортировки результатов
- return: определяется результат, который следует отобразить

Для начала надо декларировать документ XML. Пусть мой документ называется “stu”:

```
declare @stu xml;
```

Затем указывается сам документ:

```
set @stu= ''
```

Окончательно запрос метода .query выглядит так:

```
declare @stu xml;
set @stu='<root>
<Osakonna Nimi="fyysika">
  <yli ylikool="ty">
    <tudengi_nimi>kristjan</tudengi_nimi>
    <tudengi_perekonnanimi>komm</tudengi_perekonnanimi>
```

```
  </yli>
</Osakonna>
<Osakonna Nimi="informaatika">
  <yli ylikool="tly">
    <tudengi_nimi>ants</tudengi_nimi>
    <tudengi_perekonnanimi>kokk</tudengi_perekonnanimi>
  </yli>
  <yli ylikool="tly">
    <tudengi_nimi>margret</tudengi_nimi>
    <tudengi_perekonnanimi>hunt</tudengi_perekonnanimi>
  </yli>
</Osakonna>
<Osakonna Nimi="matemaatika">
  <yli ylikool="tty">
    <tudengi_nimi>liis</tudengi_nimi>
    <tudengi_perekonnanimi>aed</tudengi_perekonnanimi>
  </yli>
</Osakonna>
</root>'

select @stu.query('for $tudengi_nimi in /root/Osakonna/yli/tudengi_nimi
      return string($tudengi_nimi)
')
```

Проходя по заданному пути до нужного мне элемента, данный запрос выводит все имена студентов, независимо от университета или факультета:

Если из последней строчки убрать “string()”, то результат будет в виде элементов. Путем сортировки по узлам, можно указать конкретно интересующий меня факультет или университет. Например, хочу знать имя первого студента в списке факультетов второго узла, т.е. “füüsika”:

```
/root/Osakonna[2]/yli[1]/tudengi_nimi
```

Можно оставить сортировку только по факультету, тогда результатом будет список всех студентов с этого факультета. Данный запрос

```
select @stu.query('
      for $tudengi_nimi in /root/Osakonna/yli/tudengi_nimi
      return
        <root>
         <Osakonna>
            <yli>
               <tudengi_nimi>{data($tudengi_nimi)}</tudengi_nimi>
            </yli>
         </Osakonna>
        </root>
')
```

выполняет ту же функцию, но результат представляет в виде XML документа:

```
<root>
  <Osakonna>
    <yli>
      <tudengi_nimi>kristjan</tudengi_nimi>
    </yli>
  </Osakonna>
</root>
<root>
  <Osakonna>...
</root>
```

“Предложение **let** может быть использовано для именования повторяющихся выражений, на которые можно ссылаться с помощью ссылок на переменные. В SQL Server 2008 выражение, присвоенное переменной let, будет вставляться в запрос каждый раз, когда эта переменная будет упоминаться в запросе. Это означает, что инструкция будет выполнена столько раз, сколько было ссылок на выражение.”[4]

Однако, как оказалось, это предложение не поддерживается в SQL Sever 2005. Хотя в SQL Server 2008 сделаны изменения и с помощью “let” предложения можно объявлять переменные. В 2005 же для объявления переменных используется только “for”.

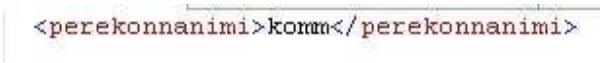
Where создает ограничения на выбранные элементы и аттрибуты для упрощения поиска.

[4] http://msdn.microsoft.com/ru-ru/library/ms190945(SQL.90).aspx 19.02.2009

```
select @stu.query('
      for $yli in /root/Osakonna/yli
            where $yli/tudengi_nimi="kristjan"
            return
                  <perekonnanimi>
                        {$yli/tudengi_perekonnanimi/text()}
                  </perekonnanimi>

')
```

Результат фильтрации предложением “where”:



Order by ()- сортирует результаты запроса по конкретным условиям. В последнем примере я решил показать работу всех предложений вместе. Допустим у нас в списке имеется несколько человек с фамилией “Pomm”. Мне нужно найти конкретного студента по его университету. Для этого декларирую “$yli”, фильтрую поиск ограничением фамилии, и сортирую университеты в алфавитном порядке, при этом указывая, что результат должен выглядеть в виде XML документа.

```
select @stu.query('
      for $yli in /root/Osakonna/yli
            where $yli/tudengi_perekonnanimi="pomm"
            order by $yli/@ylikool
            return
             <ylikooli nimi="{data($yli/@ylikool)}">
                  <nimi>
                        {$yli/tudengi_nimi/text()}
                  </nimi>
                  <perekonnanimi>
                        {$yli/tudengi_perekonnanimi/text()}
                  </perekonnanimi>
             </ylikooli>
```

```
')
```

Результат запроса:

```
<ylikooli nimi="tly">
    <nimi>vahur</nimi>
    <perekonnanimi>pomm</perekonnanimi>
</ylikooli>
<ylikooli nimi="tty">
    <nimi>viljar</nimi>
    <perekonnanimi>pomm</perekonnanimi>
</ylikooli>
<ylikooli nimi="ty">
    <nimi>andrus</nimi>
    <perekonnanimi>pomm</perekonnanimi>
</ylikooli>
```

Параметр “descending”, прописанный в конце предложения “order by”, сортирует университеты в убывающем порядке. А параметр “local-name” сортирует результат запроса по первому дочернему элементу элемента <yli>

```
order by local-name($yli)
```

Поэтому в моем случае сортировка идет по именам студентов в алфавитном порядке, не учитывая университеты.

Всегда следует проверять написание путей, элементов и аттрибутов на отсуствие пробелов и с учетом регистра шрифта!

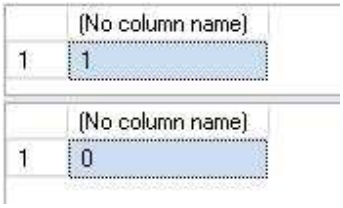
В разных источниках имеется разная информация насчет поддержки “order by” SQL Server’ом 2005.

5.2. .exist()

Этот метод проверяет XML документ на наличие узла. Если узел существует, выдается ответ с Boolean значением 1, в противном случае - 0. Я проверил свой документ, имеется ли в нем имя “kristjan” и университет под названием “ebs”.

```
select @stu.exist('/root/Osakonna/yli/tudengi_nimi[.="kristjan"]');
select @stu.exist('/root/Osakonna/yli[@ylikool="ebs"]')
```

Имя имеется, а вот университета с таким именем нет, поэтому результат:



	(No column name)
1	1

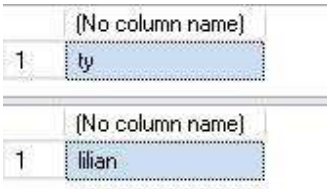
	(No column name)
1	0

5.3. .value()

.value() метод выполняет почти ту же функцию что и query(), но value() метод выдает значение определенного элемента или аттрибута в типах varchar или integer.

```
select @stu.value('(/root/Osakonna/yli/@ylikool)[1]','varchar(3)');
select @stu.value('(/root/Osakonna/yli/tudengi_nimi)[3]',
      'varchar(20)')
```

Результат:



	(No column name)
1	ty

	(No column name)
1	lilian

5.4. .nodes()

nodes() метод используется для извлечения данных из XML документа, а также для создания подузлов.

```
select uni.yli.query('.') as result from
      @stu.nodes('/root/Osakonna')uni(yli);
select uni.yli.value('@ylikool', 'varchar(3)')from
      @stu.nodes('/root/Osakonna/yli') as uni(yli)
```

Первый запрос разбивает XML документ на составные. XPath указывает, с какого места начинается разделение. В данном случае пропал главный элемент <root> и у каждой части главным элементом стал элемент <Osakonna>. Следовательно, каждый новый документ представляет из себя список отдельного университета, что очень удобно при работе с большим количеством учебных заведений и студентов.

```
<Osakonna Nimi="fyysika">
  <yli ylikool="ty">
    <tudengi_nimi>andrus</tudengi_nimi>
    <tudengi_perekonnanimi>pomm</tudengi_perekonnanimi>
  </yli>
  <yli ylikool="ty">
    <tudengi_nimi>kristjan</tudengi_nimi>
    <tudengi_perekonnanimi>komm</tudengi_perekonnanimi>
  </yli>
  <yli ylikool="ty">
    <tudengi nimi>lilian</tudengi nimi>
```

Второй запрос с помощью указанного пути XPath, нашел нужный элемент <yli>, аттрибут которого записывается в таблицу.

	(No column name)
1	ty
2	ty
3	ty
4	ty
5	tly

5.5. .modify()

Метод создан, чтобы манипулировать хранимыми в таблице XML данными.

Состоит из двух положений:

INSERT

DELETE

Создание нового узла, путем определения его структуры и XPath:

```
set @stu.modify('insert
      <aasta>2003</aasta>
      into (/root/Osakonna/yli)[1]
')select @stu
```

Результат:

```
<Osakonna Nimi="fyysika">
  <yli ylikool="ty">
    <tudengi_nimi>kristjan</tudengi_nimi>
    <tudengi_perekonnanimi>komm</tudengi_perekonnanimi>
    <aasta>2003</aasta>
  </yli>
```

Удалить узел можно с помошью этого запроса:

```
set @stu.modify('delete /root/Osakonna/yli/aasta
')
select @stu
```

6. Задания для самостоятельной работы:

Создать таблицу для типизированного , содержащую в себе список городов.

Занести в таблицу названия городов и страны.

Создать таблицу, содержащую в себе страны, материк и национальность страны.

Создать таблицу, содержащую национальности, количество жителей страны, количество жителей города.

Вывести ввиде данные таблицы, где жителей больше чем 1000000.

Вывести ввиде , где имеютса название города, континента и число жителей этого города.

Проверить существует ли страна под названием *****, если да, то добавить в имя президента.

Найти национальность страны, в городе которой количество жителей меньше чем 50000.

Применить к своей таблице все функции и методы.

Заключение

Используя XML как встроенный тип данных, приложения могут хранить документы сложной или заранее неизвестной структуры. SQL Server 2005 имеет большие возможности по поиску внутри данных XML, благодаря типизированию XML-схемами и поддержке языка запроса XQuery.

В данной работе я хотел показать насколько просто и эффективно использование XML в SQL сервере. Использование вышеупомянутого XQuery с его методами типа данных, запросами управления данными, а также применение FOR XML и OPENXML, в комбинации с T-SQL позволяют быстро и удобно оперировать документами, таблицами и их данными.

В работе я подробно рассмотрел эти конструкции, использовал множество примеров, по которым без труда можно узнать принцип работы симбиоза всемирноизвестных стандартов. И все таки для профессиональной работы лучше всего использовать полноценные версии сервера и графического приложения, ввиду ограничений и отсуствия некоторых возможностей и утилит версии Express Edition.

Ülevaade

Selle töö põhimõtteks oli tutvustada Teid XML'i tööga SQL Server'is 2005. Mina õppisin valmistama XML Schema Collections ja siduma neid XML Data Type'ga. See kindlustab valideerimist, Typed XML, ja paremat sooritamist, kui tegemist on XQuery'ga.

XQuery on üks XML andmete päringu meetodeid, mis annab kasutajale võimaluse saada andmeid XML baasist ja vaadata tabeleid just nagu teist XML andmeressurssi. Mina õppisin selle operaatoreid, FLWOR väljendeid. Olen kirjeldanud selle väljenduse clauses näidiste abil. XQuery annab kasutajale võimaluse manipuleerida ja XML'ide dokumentidest andmeid välja võtta. Samuti XQuery võtab informatsiooni andmebaasist välja, kasutades võrgu ressurse, valib ja transformeerib andmeid XHTML'i, internetti panemiseks.

FOR XML clause annab võimalust transformeerida andmed XML'i dokumendisse. Selleks, et avada XML dokumenti peab kasutama OPENXML avaldist, aga esiteks peab dokument olema ette valmistatud. Seda annab teha „sp_xml_preparedocument" funktsioon.

Kõik töö eesmärgid olid saavutatud. Minu näidiste kasutamiseks peate Teie installeerima SQL Server 2005 Express Edition ja SQL Server Management Strudio Express oma arvutile. Need versioonid on tasuta kasutamiseks ja te saate neid proovida ilma registreerimata ega maksmata.

XML'i kasutamine SQL Server'is on juba laialt levinud ja palju lihtsustab andmetega seotud tööd. XML'i ja T-SQL'i funktsioonide kasutamine on vajalik ja asjalik, ja see teema peab ülikoolide õppekavas olema. Loodan, et minu töö annab mingi abi õppimisel.

Использованая литература

1. Nielsen, P. (2006). SQL Server 2005 Bible. Indianapolis: Wiley Publishing, Inc. ISBN-10: 0764542567. ISBN-13**:** 978-0764542565.
2. Vieira, R. (2006). Professional SQL Server 2005 Programming. Indianapolis: Wrox. ISBN-10**:** 0764584340. ISBN-13**:** 978-0764584343.
3. Microsoft TechNet. http://technet.microsoft.com/en-us/sqlserver/default.aspx 19.02.2009
4. Ширшов, А. (2004). Использование XML совместно с SQL. Новые возможности Microsoft SQL Server 2005 (YUKON). http://www.rsdn.ru/article/db/xmlsql3.xml#EQLAE . 23.02.2009