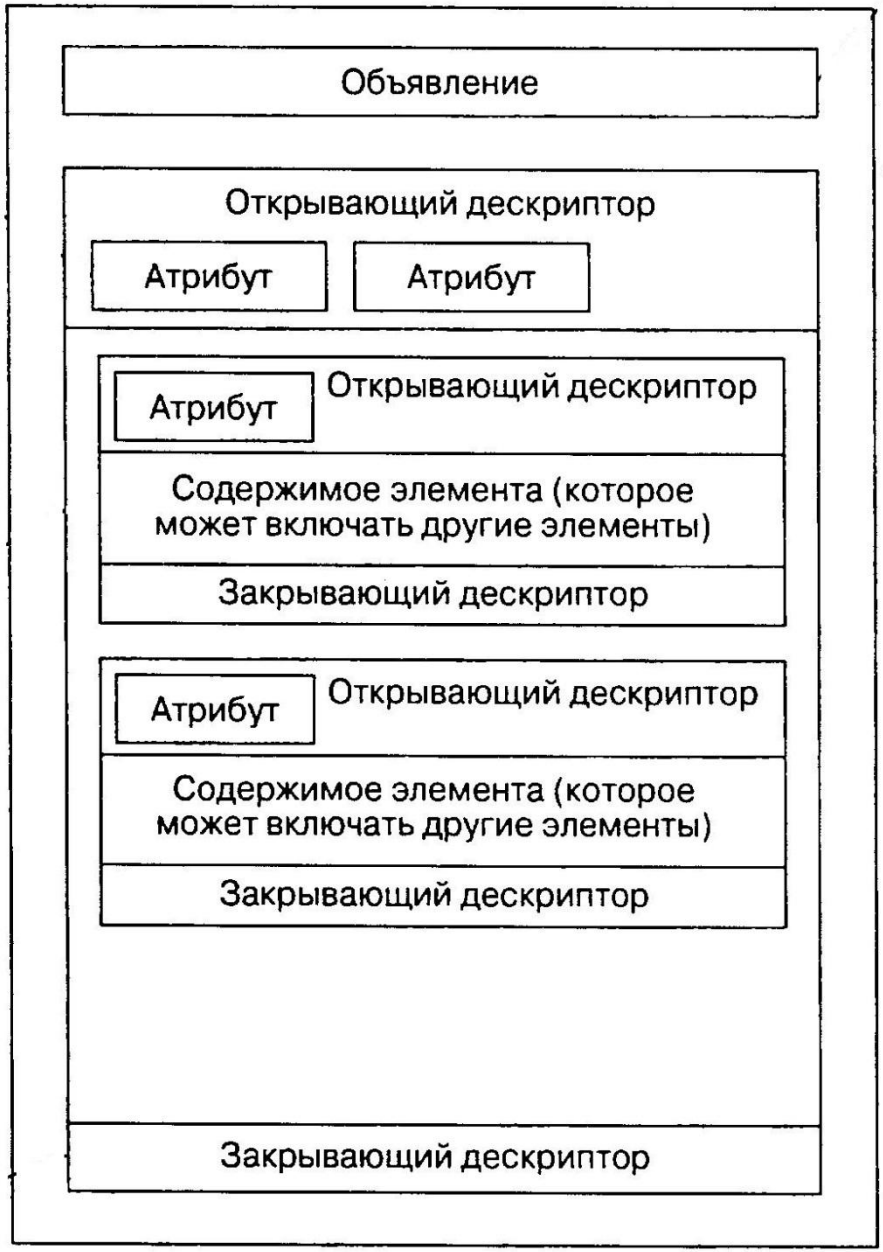
Основные сведения о языке XML

Язык XML — это основа информационных технологий. Как и HTML, язык XML основан на использовании текстовой разметки оба этих языка были созданы на основе одного и того же оригинального языка, называемого SGML.

Важным свойством документов XML является то, что они легко поддаются преобразованию.

Документ XML



Части документа XML

- ***Документ***

Документом XML называется объект самого высокого уровня, который охватывает все представленное с его помощью содержимое,
от первого символа до последнего.
Если речь идет о документе XML, то под этим
подразумевается и структура, и информационное наполнение данного конкретного документа XML.

- ***Объявление документа***

Объявление документа является необязательным, но на практике его всегда следует включать в документ. Если в документе предусмотрено использование объявления, то это объявление должно находиться в документе на самом первом месте.

Объявление представляет собой специальный дескриптор, который начинается с вопросительного знака (указывающего на то, что этот дескриптор является директивой препроцессора), за которым следует ключевое слово xml:

<?xml version="1.0"?>

В этом объявлении имеется один обязательный атрибут (атрибут — это средство дополнительного описания элемента) — **version.**
В объявление можно включить дополнительные атрибуты — **encoding** и **standalone**. Атрибут **encoding** содержит обозначение типа символьного набора, который используется в документе XML. Атрибут **standalone** сообщает, является ли документ XML независимым.

Пример

<?xml version="1.0" encoding="WINDOWS-1251" standalone="true"?>

В спецификации XML строго запрещено использование имен элементов, которые начинаются с букв xml.

- ***Элементы (теги) документа***

Элементы обозначаются именами и служат для непосредственного представления информации, хранящейся в документе. С помощью элементов может быть выражена практически любая информация. В состав элемента входит согласованная пара дескрипторов, состоящая из начального и конечного дескрипторов, которые называют также открывающим и закрывающим дескрипторами.

Открывающий дескриптор начинается с открывающей угловой скобки (<), содержит имя, а также, возможно, несколько атрибутов, за которыми следует закрывающая угловая скобка (>):
<ATagForANormalElement >

Исключением из этого правила являются элементы, содержащие в себе признак закрытия; в этом случае закрывающей угловой скобке начального дескриптора предшествует символ /, а конечный дескриптор отсутствует:
<AselfClosingElement/>

Конечные дескрипторы содержат точно такое же имя, как и начальные дескрипторы (в котором также учитывается регистр), но в конечном дескрипторе перед именем элемента должна находиться косая черта (/).

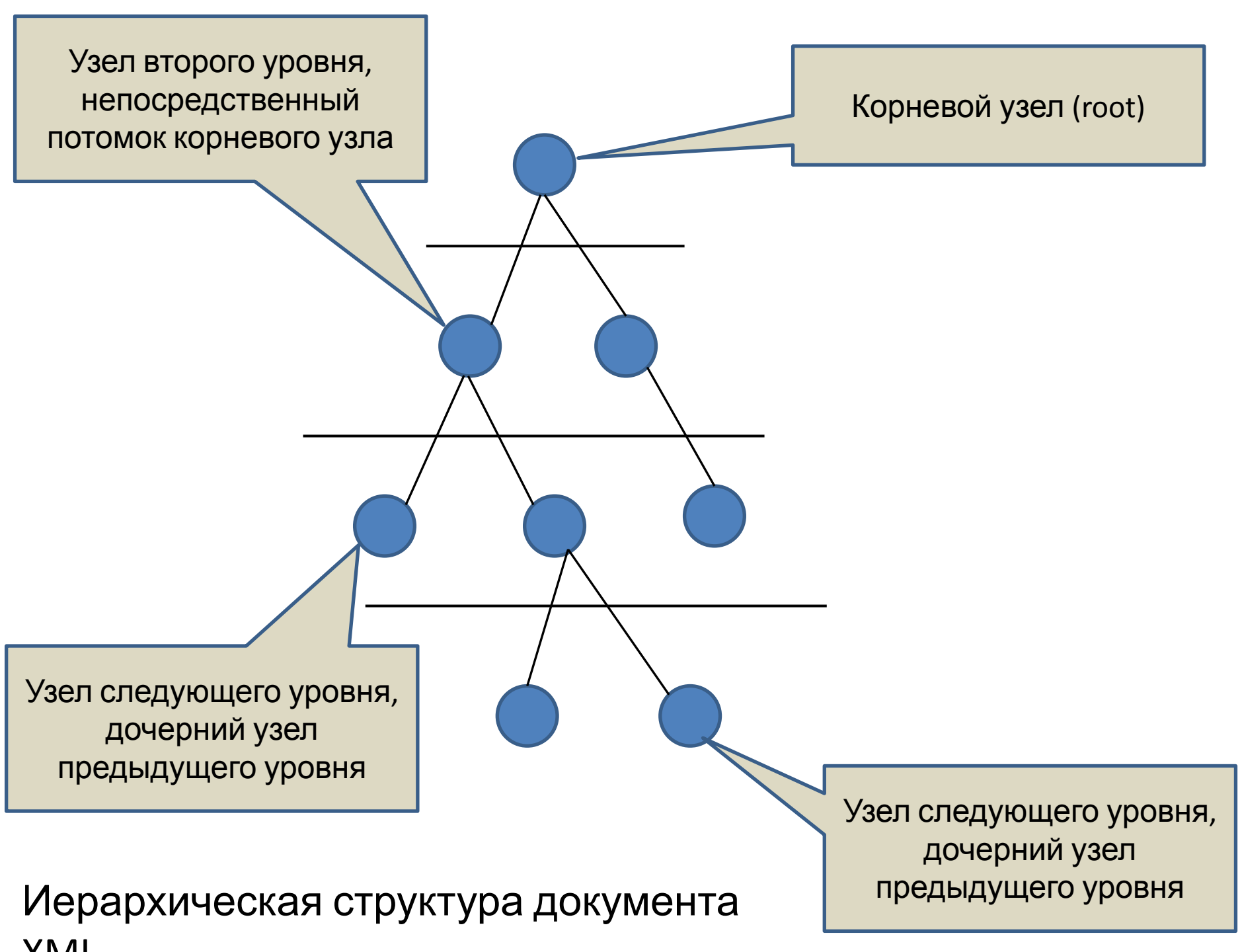
<ATagForANormalElement >
  (Здесь может находится информация)
</ATagForANormalElement > <==
Закрывающий дескриптор.

Элементы могут также включать атрибуты в составе открывающего (но не закрывающего) дескриптора элемента. Наконец, элементы могут включать другие элементы, но в таком случае внутренний элемент должен быть закрыт прежде, чем будет закрыт внешний элемент:

```
<OuterElement>
    <InnerElement>
    </InnerElement>
 </OuterElement>
```

- ***Узлы***

В связи с тем что элементы в документе XML должны подчиняться строгим пра вилам вложенности, между элементами естественным образом формируются отношения вложенности, которые образуют иерархическую структуру, имеющую древовидную форму. Каждая конечная точка связи в этом дереве называется узлом.



Иерархическая структура документа XML

- **Корневой узел**

Корневой узел представляет собой элемент, который содержит все прочие элементы в документе, в том числе элемент самого документа. Каждый документ XML имеет один (и только один) корневой узел.

Документ XML и фрагмент XML (соответствующий поддереву иерархического дерева документа) принципиально отличаются тем, что документ XML имеет корневой узел. Ни один фрагмент XML не может рассматриваться как равнозначный целому документу, поскольку в нем отсутствует корневой узел.

- ***Атрибуты***

Атрибуты могут быть представлены только в контексте элемента. Атрибуты при меняются как средство дополнительного описания элемента и находятся в пределах границ открывающего дескриптора элемента:

<SomeElement MyFirstAttribute= "Привет" MySecondAttribute="25">

Возможно, что-то еще из XML

</SomeElement>

Независимо от того, к какому типу данных относится информация, представленная в виде значения атрибута, это значение должно быть заключено в парные одинарные или двойные кавычки.

Формально правильный документ

Основным требованием к любому документу XML является то, чтобы он соответствовал определению формально правильного документа. Определение формально правильного документа XML регламентирует, какие элементы могут входить в состав документа, какую структуру он должен иметь и из каких частей состоять.

Полный перечень на сайте www.w3.org/TR/REC-xml консорциума W3C.

- Каждый документ XML должен иметь уникальный корневой узел.
- Каждому открывающему дескриптору должен соответствовать закрывающий дескриптор с таким же именем (в котором учитывается регистр); закрывающий дескриптор может отсутствовать, только если открывающий дескриптор содержит признак закрытия в самом себе.

- Не допускается, чтобы за открывающим дескриптором одного элемента непо средственно следовал закрывающий дескриптор другого элемента.

- В информационном наполнении элемента не допускается непосредственно использовать символы, которые предназначены для разметки структуры документа XML и его интерпретации с помощью синтаксического анализатора XML.

Если необходимо представить любой из таких специальных символов, то вместо него должна использоваться управляющая последовательность (которая снова преобразуется в исходный символ во время выборки информационного наполнения элемента).

Пример формально правильного документа.

```
<?xml version="1.0" encoding="UTF-8"?>
<ThisCouldBeCalledAnything>
   <AnElement>
      <AnotherElement AnAttribute="Some Value">
<AselfClosingElement
AnAttributeThatNeedsASpecialCharacter= “Д&amp;quot;
Артаньян"/>
      </AnotherElement>
   </AnElement>
 </ThisCouldBeCalledAnything>
```

Примеры кода XML

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <phone Family=”Иванов” Type=”Домашний”
Number=”123-45-67” />
  <phone Family=”Иванов” Type=”Мобильный 1”
Number=”123-45-67-89” />
  <phone Family=”Иванов” Type=”Мобильный 2”
Number=”123-45-67-98” />
  <phone Family=”Иванов” Type=”Рабочий”
Number=”123-45-76” />
</root>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <Abonent  Family=”Иванов” >
     < phone Type=”Домашний” Number=”123-45-67” />
     < phone Type=”Мобильный 1” Number=”123-45-67-89” />
     < phone Type=”Мобильный 2” Number=”123-45-67-98” />
     < phone Type=”Рабочий” Number=”123-45-76” />
  < /Abonent >
</root>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <Abonent  Family=”Иванов” >
     < phone Type=”Домашний” Number=”123-45-67” />
     < phone Type=”Мобильный 1” Number=”123-45-67-89”
/>
     < phone Type=”Мобильный 2” Number=”123-45-67-98”
/>
     < phone Type=”Рабочий” Number=”123-45-76” />
  < /Abonent >
  <Abonent  Family=”Петров” >
     < phone Type=”Домашний” Number=”231-45-67” />
     < phone Type=”Мобильный 1” Number=”231-45-67-89”
/>
     < phone Type=”Мобильный 2” Number=”231-45-67-98”
/>
     < phone Type=”Рабочий” Number=”231-45-76” />
```

Сравнение способов представления данных с помощью элементов и атрибутов

Нет твердых и окончательно установленных правил в отношении того, какую информацию следует представлять с помощью элементов или атрибутов. Атрибут описывает то, что является свойством элемента, к которому относится сам атрибут.

Применение атрибутов оправдано в тех ситуациях, когда некоторое значение находится связи "один к одному" с элементом и является его неотъемлемой частью. Использование элементов в большей степени оправдано, если между самим элементом и теми данными, которые он

Пространства имен

Чтобы связать с определенным пространством имен весь документ, достаточно ввести в определение корневого элемента документа ссылку на пространство имен в виде специального атрибута (именуемого xmlns). Кроме того, ссылки на пространства имен (опять-таки формируемые с помощью атрибута xmlns) могут быть введены и в другие элементы документа, если требуется распространить действие данного конкретного пространства имен только на ту часть документа, которую занимает данный элемент.

```
<?xml version="1.0" encoding="UTF-8"?>
<Schema xmlns="urn:schemas-microsoft-com:xml-data"
        xmlns:dt="urn:schemas-microsoft-com:datatypes"
        xmlns:sql="urn:schemas-microsoft-com:xml-sql"
        sql:xsl='../Customers.xsl'>
 <ElementType name="Root" content="empty"/>
 <ElementType name="Customers" sql:relation="Customers"/>
       <AttributeType name="CustomerID"/>
       <AttributeType name="CompanyName"/>
       <AttributeType name="Address"/>
       <AttributeType name="City"/>
<attribute type="CustomerID" sql:field="CustomerID"/>
  <attribute type="CompanyName" sql:field="CompanyName "/>
        <attribute type="Address" sql:field="Address"/>
        <attribute type="City" sql:field="City"/>
</ElementType>
 </Schema>
```

В документе имеются ссылки на три пространства имен. Одно из них относится к определению **XDR** (*External Data Representation - внешнее представление данных*) — международный стандарт передачи данных в Интернете), другое представляет собой пространство имен типов данных Microsoft (в этом пространстве имен представлен перечень, определяющий количество и специфику различных типов данных) и, наконец, предусмотрено специальное пространство имен SQL, предназначенное для работы со средствами интеграции XML программы SQL Server. Некоторые атрибуты (включая заданные в корневом элементе) уточняются с по мощью информации о пространстве имен (в

Средства формирования документов XML, в СУБД SQL Server

Основные направления поддержки XML:

- Поддержка нескольких методов выборки данных из обычных столбцов и оформление этих данных в формате XML.
- Обеспечение возможности непосредственного хранения данных XML в СУБД SQL Server.

- Предоставление функций запроса к данным, хранящимся в первоначальном формате XML, с помощью языка XQuery.
- Поддержка средства обеспечения целостности данных, хранящихся в формате XML, с помощью схем XML.
- Обеспечение индексации данных XML.

Выборка реляционных данных в формате XML

Конструкция **FOR XML** определена как еще одна необязательная конструкция, которая добавлена в конце существующей синтаксической структуры оператора SELECT языка T-SQL.

SELECT <column list>
[FROM <source table (s)>j
[WHERE restrictive condition>]
[GROUP BY <column name >
[HAVING Restrictive condition based on the GROUP BY results>]
[ORDER BY <column list>]
[FOR XML {RAW|AUTO|EXPLICIT [, XMLDATA] [, ELEMENTS]
[, BINARY base64] [PATH]
[OPTION (<query hint>, [, ...n])]

В конструкции FOR XML предусмотрено несколько различных начальных опций, позволяющих указать, как должны быть отформатированы результаты запроса в виде кода XML.

- Опция **RAW**. Ключевое слово RAW указывает, что каждая строка данных в результирующем наборе должна быть возвращена как один элемент данных. Элемент получает имя "row", а каждое поле строки оформляется в виде атрибута элемента "row".

- Опция **AUTO**. При использовании опции AUTO каждый элемент получает обо значение, взятое либо из имени таблицы, либо из псевдонима имени таблицы, которая является источником данных.

При использовании опции AUTO поддерживается также дополнительная опция ELEMENTS, позволяющая указать, что данные полей должны быть представлены в виде элементов, а не атрибутов.

- Опция **EXPLICIT**. Опция EXPLICIT требует применения наиболее сложных синтаксических конструкций при оформлении запроса, но в конечном итоге позволяет получить наибольший контроль над тем, как будут выглядеть данные в коде XML.

Возможности опции EXPLICIT перекрываются возможностями, предусмотренными в опции PATH, поэтому поддержка опции EXPLICIT осуществляется главным образом для обеспечения обратной совместимости.

- Опция **PATH** [**('***ElementName***')**]. Создает упаковщик элементов <строки> для каждой строки в результирующем наборе. Для упаковщика элементов <строки> можно дополнительно задать имя элемента. Использование директивы PATH дает более простой способ написания запросов, чем написание запросов с помощью директивы EXPLICIT.

Необходимо учитывать, что ни одна из указанных опций не обеспечивает формирования обязательного корневого элемента.

Кроме четырех основных опций форматирования, предусмотрены также другие необязательные параметры, позволяющие дополнительно уточнять формат выходных данных, предоставляемых в СУБД SQL Server при выполнении запроса с ключевым словом FOR XML.

- Опция XML DATA. Эта опция служит для СУБД SQL Server указанием на то, что формируемым результатам должно предшествовать определение схемы XML. Схема определяет структуру документа (в том числе состав используемых типов данных), а также правила, которым должны соответствовать данные XML.

- Опция ELEMENTS. Эта опция является применимой, только если используется также опция форматирования AUTO. Опция ELEMENTS служит для СУБД SQL Server указанием на то, что данные полей в возвращаемых результатах должны быть оформлены в виде вложенных элементов, а не атрибутов.

- Опция BINARY BASE64. Эта опция является для СУБД SQL Server указанием на то, что содержимое всех полей с двоичными данными (binary, varbinary, image varchar(max), nvarchar(max) ,varbinary(max)) должно быть закодировано в формате BASE64. Если используется опция AUTO, то опция BINARY BASE64 вводится в действие по умолчанию.

- Опция TYPE. Эта опция рассматривается в СУБД SQL Server в качестве указания на то, что возвращаемые результаты должны быть обозначены как относящиеся к типу данных xml, а не к применяемому по умолчанию символьному типу данных Unicode.

- Опция ROOT [**('***RootName***')**]. С помощью данной опции можно указать, что корневой элемент должен быть сформирован в СУБД SQL Server, поэтому разработчику не нужно об этом заботиться. В таком случае можно либо явно задать имя корневого элемента, либо предусмотреть использование имени корневого элемента, заданного по умолчанию (root).

- Опция XMLSCHEMA [**('***TargetNameSpaceURI***')**]. Возвращает встроенную XSD-схему. При задании указанной директивы, возвращающей заданное пространство имен схемы, дополнительно можно задать URI целевого пространства имен.

- Опция XSINIL. Директива ELEMENTS формирует XML, в котором каждое значение в столбце соответствует элементу XML. Если этот значение в столбце имеет значение NULL, элемент не добавляется. Указав в директиве ELEMENTS необязательный параметр XSINIL, можно запросить, чтобы для значений NULL тоже создавались элементы. В этом случае атрибут **xsi:nil** этого элемента имеет значение TRUE для каждого значения NULL в столбце. Данный параметр может быть указан только в директиве ELEMENTS.

- Опция ABSENT. Указывает, что соответствующие XML-элементы для столбцов со значениями NULL к XML-результату не добавляются. Указывайте данный параметр только с директивой ELEMENTS.

Режим EXPLICIT

Режим EXPLICIT преобразует набор строк, получаемый в результате выполнения запроса, в XML-документ. Для того чтобы режим EXPLICIT создал XML-документ необходимо написать запрос SELECT для создания набора строк, **универсальной таблицы**, имеющей определенный формат, так чтобы логика обработки могла создать желаемый XML.

Запрос должен создавать следующие два столбца метаданных:

- первый столбец должен предоставлять номер тега текущего элемента (целочисленного типа), и этот столбец должен иметь имя **Tag**. В запросе должен быть указан уникальный номер тега для каждого элемента, который будет создан из набора строк;

- второй столбец должен задавать номер тега для родительского элемента, и этот столбец должен иметь имя **Parent**. Таким образом, столбцы Tag и Parent предоставляют сведения об иерархии.

Значения этих столбцов метаданных вместе со сведениями в именах столбцов используются для создания желаемого XML. Имена столбцов в запросе должны задаваться особым образом. Значение 0 или NULL в столбце **Parent** указывает на то, что у соответствующего элемента нет родителя. Такой элемент добавляется в XML в качестве элемента верхнего уровня.

Столбцы Tag и Parent имеют вполне определенное назначение, а имена остальных столбцов включают несколько фрагментов метаданных. Единственным обозначением, позволяющим определить, где заканчивается один фрагмент метаданных и начинается другой, служит восклицательный знак (!).
Для именования столбцов применяется примерно такой формат:
<element name>!<tag>![<attribute name>]
[!{element|hide|ID|IDREFI
REFS|xml|xmltext|cdata}]

Директивы id, idref и idrefs

Ни одна из директив id, idref и idref s не выполняет каких-либо действий, если в сочетании с ними не используется также опция XMLDATA (которая должна следовать за опцией EXPLICIT в конструкции FOR XML).

С их помощью формируются документы, предназначенные для проверки по схеме, которая содержит соответствующие объявления.
Директивы вводят дополнения к схеме, позволяющие провести проверку определенных характеристик документа, но само проведение такой проверки без схемы является невозможным.

Режим PATH

Режим PATH является простым способом смешивания элементов и атрибутов. Режим PATH является также простым способом создания дополнительных вложенных объектов для отражения сложных свойств. В режиме PATH имена или псевдонимы столбцов обрабатываются как выражения XPath.

XPath

XPath (XML Path Language) — язык запросов к элементам XML-документа. Разработан для организации доступа к частям документа XML в файлах трансформации XSLT и является стандартом консорциума W3C. XPath призван реализовать навигацию по DOM в XML.

DOM (Document Object Model — «объектная модель документа») — это не зависящий от платформы и языка программный интерфейс, позволяющий получить доступ к содержимому HTML, XHTML и XML-документов, а также изменять содержимое, структуру и оформление таких документов.

В XPath используется компактный синтаксис, отличный от принятого в XML.

ФУНКЦИЯ OPENXML

OPENXML — это функция для работы с наборами строк, которая позволяет открыть переданную ей строку. Это означает, что документ XML можно использовать в операции соединения или даже сделать его источником входных данных, применив операцию INSERT. . . SELECT или SELECT INTO.

Основное различие между OPENXML и другими функциями состоит в том, что в сочетании с этой функцией необходимо использовать целый ряд системных хранимых процедур для подготовки документа, а затем для очистки памяти после завершения обработки документа.

Для подготовки документа к работе служит процедура sp_xml_preparedocument, которая перемещает переданную ей строку в память и выполняет предварительный синтаксический анализ этой строки для достижения оптимальной производительно сти запроса.

sp_xml_preparedocument @hdoc = <integer variable> OUTPUT,
[, @xmltext = <character data>]
[, @xpath_namespaces = <url to a namespace>]

После вызова этой хранимой процедуры и сохранения дескриптора документа можно приступить к использованию функции OPENXML. Синтаксис вызова функции :

OPENXML(<handle>,<XPath to base node>[, <mapping flags>]) [WITH (<Schema Declaration|<Table Name>)]

- **<handle>** - дескриптор представляет собой целочисленное значение, которое должно быть получено в виде выходного параметра после вызова процедуры *sp_xml_preparedocument*.
- **<XPath to base node>** - представляющий собой обозначение пути к узлу, который должен служить в качестве начальной точки для всех запросов.

- **<Schema Declaration>** - позволяет ссылаться на любые части документа XML с помощью указания пути перемещения относительно базового узла, заданного в этом параметре.
- **<mapping flags>** - позволяют указать, должно ли быть в результатах вызова функции OPENXML отдано предпочтение элементам или атрибутам. Возможные значения (0, 1, 2, 8).

ФУНКЦИЯ OPENROWSET

Содержит все необходимые сведения о соединении, которые требуются для доступа к удаленным данным источника данных OLE DB. Это альтернативный метод для доступа к таблицам на связанном сервере.

Функция OPENROWSET также поддерживает массовые операции с помощью встроенного поставщика BULK, позволяющего считывать данные из файла и возвращать их в виде набора строк.

```
OPENROWSET
( { 'provider_name', { 'datasource';'user_id';'password'
   | 'provider_string' }
  , {   [ catalog. ] [ schema. ] object
      | 'query'
    }
  | BULK 'data_file',
     { FORMATFILE ='format_file_path' [ <bulk_options> ]
     | SINGLE_BLOB | SINGLE_CLOB | SINGLE_NCLOB }
} )
```

<bulk_options> ::=
[, CODEPAGE = { 'ACP' | 'OEM' | 'RAW' | 'code_page' }]
[, ERRORFILE ='file_name']
[, FIRSTROW =first_row]
[, LASTROW =last_row]
[, MAXERRORS =maximum_errors]
[, ROWS_PER_BATCH =rows_per_batch]
[, ORDER ({ column [ASC | DESC] } [,...n]) [UNIQUE]