1. **Опишите структуру XML-файла.**

<?xml version="1.0" encoding="windows-1251"?>

<book category="WEB">

<title lang="en">Learning XML</title>

<author>Erik T. Ray</author>

<year>2003</year>

<price></price>

</book>

Первая строка — это XML декларация. Здесь определяется версия XML (1.0) и кодировка файла. На следующей строке описывается корневой элемент документа <book> (*открывающий тег*). Следующие 4 строки описывают дочерние элементы корневого элемента ( title, author, year, price). Последняя строка определяет конец корневого элемента </book> (*закрывающий тег*).

Документ XML состоит из *элементов* (elements). Элемент начинается *открывающим тегом* (start-tag) в угловых скобках, затем идет *содержимое* (content) элемента, после него записывается *закрывающий тег* (end-teg) в угловых скобках.

Информация, заключенная между тегами, называется содержимым или значением элемента: <author>Erik T. Ray</author>. Т.е. элемент author принимает значение Erik T. Ray. Элементы могут вообще не принимать значения.

Элементы могут содержать *атрибуты*, так, например, открывающий тег <title lang="en"> имеет атрибут lang, который принимает значение en. Значения атрибутов заключаются в кавычки (двойные или ординарные).

1. **Чем валидный XML-файл отличается от правильно построенного?**

Структура XML документа должна соответствовать определенным правилам. XML документ отвечающий этим правилам называется валидным (англ. Valid — правильный) или синтаксически верным. Соответственно, если документ не отвечает правилам, он является невалидным .

Основные правила синтаксиса XML:

* Теги XML регистрозависимы — теги XML являются регистрозависимыми. Так, тег <Letter> не то же самое, что тег <letter>. Открывающий и закрывающий теги должны определяться в одном регистре.
* XML элементы должны соблюдать корректную вложенность.
* У XML документа должен быть корневой элемент — XML документ должен содержать один элемент, который будет родительским для всех других элементов. Он называется корневым элементом.
* Значения XML атрибутов должны заключаться в кавычки.

1. **Что такое DTD? Опишите структуру DTD файла.**

**Язык схем DTD** (DTD schema language) — компьютерный язык, который используется для записи фактических синтаксических правил метаязыка SGML и расширяемого языка разметки XML. С момента его внедрения другие языки схем для языков разметки, такие как XML Schema и RELAX NG, обзавелись дополнительной функциональностью.

1. **Что такое схема? Опишите структуру файла схемы.**

Элементы

Элементы являются основными строительными блоками как XML, так и HTML документов.

В качестве примеров HTML элементов можно привести тег **<body>** или тег **<table>**. Примерами XML элементов могли бы стать теги **<note>** и **<message>**". Элементы могут содержать текст, другие элементы или быть пустыми. Примерами пустых HTML элементов являются теги **<hr>**, **<br>** и **<img>**.

Примеры:

<body>какой-то текст</body>

<message>какой-то текст</message>

Атрибуты

Атрибуты предоставляют дополнительную информацию об элементах.

Атрибуты всегда помещаются внутри открывающего тега элемента. Атрибуты всегда определяются в виде пар имя/значение. У следующего элемента **<img>** есть дополнительная информация об исходном файле:

<img src='computer.gif' />

Имя элемента - "*img*". Имя его атрибута - "*src*". Значение этого атрибута — "*computer.gif*". Так как сам элемент является пустым, он закрывается символом " /".

Сущности

Некоторые символы в XML имеют особое значение, как, например, знак "меньше чем" (<), который определяет начало тега XML.

Большинству из вас известна HTML сущность "&nbsp;". Эта сущность "неразрывного пробела" используется в HTML, чтобы вставить в код документа дополнительный пробельный символ. Во время анализа и разбора документа парсером сущности раскрываются.

Следующие сущности предопределены в XML:

| **Сущность** | **Символ** |
| --- | --- |
| &lt; | < |
| &gt; | > |
| &amp; | & |
| &quot; | " |
| &apos; | ' |

PCDATA

PCDATA означает анализируемые символьные данные.

Символьные данные это текст, который находится между открывающим и закрывающим тегами XML элемента.

PCDATA — это текст, который будет анализироваться парсером. Т.е. парсер будет проверять этот текст на наличие сущностей и другой разметки.

Теги внутри такого текста будут восприняты, как вложенная разметка, а сущности будут раскрыты.

Тем не менее, анализируемые символьные данные не должны содержать символы **&**, **<** или **>**. Они должны быть представлены соответствующими сущностями **&amp;**, **&lt;** и **&gt;**.

CDATA

CDATA означает [неанализируемые символьные данные](https://msiter.ru/tutorials/uchebnik-xml-dlya-nachinayushchih/cdata-tekstovye-dannye-xml" \o "CDATA - текстовые данные XML).

CDATA — это текст, который не будет анализироваться парсером. Теги внутри такого текста не будут восприняты, как вложенная разметка, а сущности не будут раскрыты.

1. Поясните разницу между различными видами представления реляционных данных в XML в SQL Server.

SQL Server поддерживает несколько вариантов хранения XML-данных.

**Естественное хранение в виде типа xml**

Данные при этом хранятся во внутреннем представлении, которое обеспечивает неизменность XML-содержимого данных. Это внутреннее представление включает в себя сведения об иерархии контейнеров, порядке документов и значений элементов и атрибутов. Точнее говоря, при этом обеспечивается неизменность InfoSet-содержимого XML-данных. Дополнительные сведения об информационном наборе см. в разделе [http://www.w3.org/TR/xml-infoset](https://go.microsoft.com/fwlink/?LinkId=48843). InfoSet-содержимое не всегда идентично текстовым XML-данным, потому что следующая информация при этом не сохраняется: несущественные пробелы, порядок атрибутов, префиксы пространств имен и XML-декларация.

Для типизированного (то есть связанного с XML -схемой) типа данных xml модуль проверки после обработки схемы (PSVI) добавляет в информационный набор данные о типах и кодирует их во внутреннее представление. Это значительно ускоряет синтаксический анализ. Дополнительные сведения см. в спецификациях XML-схемы W3C на страницах [http://www.w3.org/TR/xmlschema-1](https://go.microsoft.com/fwlink/?LinkId=48881) и [http://www.w3.org/TR/xmlschema-2](https://go.microsoft.com/fwlink/?LinkId=4871).

**Сопоставление XML-данных и данных, хранящихся в реляционном формате**

Используя аннотированную схему (AXSD), можно разбить XML на столбцы одной или нескольких таблиц. Это обеспечивает правильность данных на реляционном уровне. В результате гарантируется сохранность иерархической структуры данных, хотя порядок элементов не учитывается. Схема не может быть рекурсивной.

**Хранение больших объектов, [n]varchar(max) и varbinary(max)**

При этом хранится идентичная копия данных. Это полезно в приложениях специального назначения, например в приложениях, обрабатывающих юридическую документацию. Большинству приложений точная копия данных не нужна — им хватает XML-содержимого (правильности элементов InfoSet).

1. **Какие виды индексов для XML существуют в SQL Server?**

XML-индексы разделяются на следующие категории.

* **Первичный XML-индекс**
* **Вторичные XML-индексы.**

Первым индексом, создаваемым для столбца типа данных xml , должен быть первичный XML-индекс. При наличии первичного XML-индекса поддерживаются вторичные индексы трех типов: PATH, VALUE и PROPERTY. Эти вторичные индексы могут способствовать повышению производительности выполнения разных типов запросов.

При создании первичного XML-индекса индексируются все теги, значения и пути в экземплярах XML, хранимых в XML-столбце. Чтобы создать первичный XML-индекс, таблица, содержащая соответствующий XML-столбец, должна иметь кластеризованный индекс первичного ключа таблицы. SQL Server строки первичного XML-индекса сопоставляются строкам таблицы, в которой содержится XML-столбец.

**Ниже приведены некоторые рекомендации по созданию вторичных индексов.**

Если при работе с XML-столбцами часто используются выражения пути, вторичный XML-индекс **PATH**, скорее всего, ускорит обработку данных. Типичный пример — выполнение метода exist() для XML-столбцов в предложении WHERE инструкции Transact-SQL.

Если с использованием выражений пути извлекаются множественные значения из отдельных экземпляров XML, может принести пользу кластеризация путей в пределах каждого экземпляра XML в индекс **PROPERTY**. Этот сценарий обычно имеет место при работе с наборами свойств, когда извлекаются свойства объекта и известно значение его первичного ключа.

Если запрашиваются значения экземпляров XML, не зная имен элементов или атрибутов, содержащих эти значения, следует подумать о создании индекса **VALUE**. Как правило, это имеет место при уточняющем запросе по осям нижних уровней, например //author[last-name="Howard"], где элементы <author> могут встречаться на любом уровне иерархии. Кроме того, такая ситуация встречается при обработке запросов с символами-шаблонами (например, /book [@\* = "novel"], где в запросе выполняется поиск элементов <book>, имеющих некоторый атрибут со значением novel).

1. **Поясните назначение функций и методов EXTRACTVALUE, EXISTSNODE, GETSTRING, GETROOTELEMENT, XMLELEMENT, XMLATTRIBUTES, XMLAGG.**

The **EXTRACTVALUE** function takes as arguments an XMLType instance and an XPath expression and returns a scalar value of the resultant node. The result must be a single node and be either a text node, attribute, or element. If the result is an element, then the element must have a single text node as its child, and it is this value that the function returns.

**EXISTSNODE** determines whether traversal of an XML document using a specified path results in any nodes.

**XMLAgg**is an aggregate function. It takes a collection of XML fragments and returns an aggregated XML document. Any arguments that return null are dropped from the result.