МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение   
высшего образования

**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Кафедра вычислительная техника

**ЭССЕ по дисциплине «Автоматизация проектирования информационных систем»**

**Глава №3 «МОДЕЛИРОВАНИЕ ДАННЫХ В ФОРМАЛИЗМЕ СПЕЦИФИКАЦИИ XSD»**

Отчёт составил:  
Бакалавр гр. ИВТАСбд-42  
Сулейманов М.З.

Отчёт принял:  
Профессор кафедры ВТ  
Токмаков Г.П.

Ульяновск

УлГТУ

2024

3.1 XML КАК СПОСОБ ЛОГИЧЕСКОГО ПРЕДСТАВЛЕНИЯ ИНФОРМАЦИИ

3.1.1 XML-ПЛАТФОРМА И СРЕДСТВА ОПИСАНИЯ ЛОГИЧЕСКИХ ЧАСТЕЙ ДО КУМЕНТОВ

XML (Extensible Markup Language) был разработан для описания логической структуры документов. В отличие от HTML, который акцентируется на форматировании, XML ориентирован на строгое структурирование данных. Он предоставляет основу для создания специализированных языков разметки, таких как XML Schema, и позволяет четко разделять содержание документа и его форматирование. XML вырос из SGML (Standard Generalized Markup Language) — метаязыка для определения языков разметки, обеспечивая универсальность и гибкость при описании различных типов данных.

3.1.2 ПРОСТОЙ ПРИМЕР XML-ДОКУМЕНТА

Способность языка реализовать логическую разметку доку-XML

ментов была использована для представления данных, не зависящего от используемых информационных технологий.

Исходный документ в виде ASCII-текста. Пусть у нас имеется

документ, представленный в виде текста:

Номер клиента 12345

Имя клиента Федор Петров

Адрес клиента ул. Мира, д5, кв. 42

Телефон клиента (095) 123 45 67

Почтовый индекс клиента 123456

Лимит кредита клиента 1 000 000

Кредитоспособность клиента 5

Анализ данных документа. На основе этого документа специалист в области моделирования данных должен создать сложный элемент , содержащий простые типизированные элементы.

Номер клиента 12345 целое число

Имя клиента Федор Петров текст

Адрес клиента ул. Мира, д5, кв. 42 текст

Телефон клиента (095) 123 45 67 целое число

Почтовый индекс клиента 123456 текст

Лимит кредита клиента 1 000 000 десятичное число

Кредитоспособность клиента 5 целое число

Формирование XML-документа. Программист форматирует полученные данные в виде -документа следующим образом:

<?xml version="1.0" encoding="UTF-8"?>

<Customer>

<НомерКлиента>12345</НомерКлиента>

<ИмяКлиента>Федор Петров</ИмяКлиента>

<АдресКлиента>ул. Мира, д5, кв. 42</АдресКлиента>

<ТелефонКлиента>(095) 123 45 67</ТелефонКлиента>

<ПочтовыйИндекс>123456</ПочтовыйИндекс>

<ЛимитКредита>1 000 000</ЛимитКредита>

<Кредитоспособность>5</Кредитоспособность>

</Customer>

XML-схема документа. После представления данных о клиенте

в виде -описания и имея в наличии требования к данным этого описания, аналитик компании для проверки достоверности сведений

на предмет соответствия структуре и типам данных создает следующую схему:

<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"

elementFormDefault="qualified" attributeFormDefault="unqualified">

<xs:element name="Customer" type="CustomerType"/>

<xs:complexType name="CustomerType">

<xs:sequence>

<xs:element name="НомерКлиента" type="xs:integer"/>

<xs:element name="ИмяКлиента" type="xs:string"/>

<xs:element name="АдресКлиента" type="xs:string"/>

<xs:element name="ТелефонКлиента" type="xs:string"/>

<xs:element name="ПочтовыйИндекс" type="PostalCodeType"/>

<xs:element name="ЛимитКредита" type="xs:decimal"/>

<xs:element name="Кредитоспособность" type="xs:integer"/>

</xs:sequence>

</xs:complexType>

<xs:simpleType name="PostalCodeType">

<xs:restriction base="xs:string">

<xs:pattern value="\d{5}|\d{5}-d{4}"/>

</xs:restriction>

</xs:simpleType>

</xs:schema>

Пример иллюстрирует схему, отвечающую за проверку достоверности, типов данных и структуры документа, включающего запись о клиенте.

3.1.3 XML КАК УНИВЕРСАЛЬНЫЙ СИНТАК СИСПРЕДСТАВЛЕНИЯ ДАННЫХ

Основное преимущество XML заключается в том, что программы разных производителей могут обмениваться данными на едином языке. Это решает проблему разрозненных форматов представления информации и позволяет передавать данные между приложениями, работающими в различных средах.

Унификация достигается благодаря системе типов данных, описанной в спецификации XSD. Каждый язык программирования имеет свои определения типов и структур данных, которые указывают, сколько памяти выделяется под переменные.

Ранее разработчики были вынуждены создавать программы для конвертации данных между языками, так как каждый из них использовал уникальные механизмы для формирования и хранения информации. С появлением XML эта необходимость отпала, поскольку XML-процессоры обладают функциями преобразования данных между форматами.

Хотя отображение значений в текстовую форму может показаться неэффективным (текст требует больше памяти и времени для обработки), XML предлагает решение для интеграции разнородных данных. Программы, использующие XML, хранят данные в текстовом формате, что позволяет им легко обмениваться информацией, используя типы данных, заданные в XSD-схеме.

3.2 СРЕДСТВА МОДЕЛИРОВАНИЯ ДАННЫХ, ОПРЕДЕЛЕННЫЕ В СПЕЦИФИКАЦИИ XSD

В начале главы отмечалось, что спецификации XSD могут использоваться для моделирования данных в различных приложениях, включая реляционные базы данных, объектные представления на серверах и пользовательские интерфейсы на стороне клиента. Рассмотрим подробнее спецификацию XSD, которая может стать основой для новой парадигмы разработки корпоративных приложений.

XSD-схема — это язык, предназначенный для определения структуры XML-документов, включая их организацию и используемые типы данных. Она описывает, как определить каждый элемент в схеме и привязать к ним соответствующие типы данных. XSD-схема сама является документом XML, использующим элементы и атрибуты для выражения своей семантики.

3.2.1 ПРОСТЫЕ И СЛОЖНЫЕ ТИПЫ

Создание схемы начинается с отслеживания структуры документа и определения элементов по мере их обнаружения. Элементы, не содержащие других элементов или атрибутов, относятся к простому типу (simpleType). Например, элементы «Фамилия», «Имя», «Отчество» и «ДатаРождения» могут быть определены следующим образом:

<xs:element name="Фамилия" type="xs:string"/>

<xs:element name="Имя" type="xs:string"/>

<xs:element name="Отчество" type="xs:string"/>

<xs:element name="ДатаРождения" type="xs:date"/>

Элементы, содержащие другие элементы, определяются как сложные (complexType). Например, элемент «Адрес» может быть описан как сложный тип с использованием элемента sequence, который задает упорядоченную последовательность подэлементов.

Кроме того, в XSD-схеме можно объявлять атрибуты, которые обеспечивают альтернативный способ хранения информации в элементе.

3.2.2 КАРДИНАЛЬНОСТЬ

Язык XML Schema позволяет задавать кардинальность элементов с помощью атрибутов minOccurs (минимальное количество вхождений) и maxOccurs (максимальное количество вхождений). Для обозначения необязательного элемента minOccurs устанавливается в 0, а для неограниченного количества вхождений maxOccurs — в unbounded. Значение по умолчанию для неуказанных атрибутов — 1. Например, элемент «Отчество» может быть описан так:

<xs:element name="Отчество" type="xs:string" minOccurs="0"/>

Для регистрации двух фамилий используется:

<xs:element name="Фамилия" type="xs:string" maxOccurs="2"/>

3.2.3 ОПРЕДЕЛЕНИЕ НОВЫХ ТИПОВ

XML Schema предоставляет возможность создания новых типов данных, аналогично определению классов. Новым типам присваиваются имена, а их определения размещаются отдельно от элементов и атрибутов. Например, новый простой тип для элемента «Группа» можно определить так:

<xs:simpleType name="Группа">

<xs:restriction base="xs:string">

<xs:maxLength value="7"/>

</xs:restriction>

</xs:simpleType>

Этот тип имеет ограничение по длине, а также может включать другие фасеты, такие как length, minLength, maxInclusive и maxExclusive. Например, элемент «Группа» может иметь шаблон, ограничивающий его значение:

<xs:pattern value="[А-Я]{4}[а-я]{1}-[1-6]{2}"/>

Примеры значений: «ЭВМд-32», «АД-12», «БЭВМд-32».

Элемент enumeration позволяет ограничить простой тип набором определенных значений. Например, для элемента «Статус», который может принимать значения «Бакалавр», «Специалист» или «Магистр», это будет выглядеть так:

<xs:enumeration value="Бакалавр"/>

<xs:enumeration value="Специалист"/>

<xs:enumeration value="Магистр"/>

3.2.4 ЭЛЕМЕНТЫ СОСТАВИТЕЛИ CHOICE И ALL

В XML Schema предусмотрены два дополнительных типа элементов-составителей: choice и all.

Элемент-составитель choice позволяет выбрать один из нескольких допустимых элементов или групп элементов. Например, кафедра может быть представлена либо кодом, либо наименованием:

<xs:element name="Кафедра">

<xs:complexType>

<xs:choice>

<xs:element name="Код" type="xs:integer"/>

<xs:element name="Наименование" type="xs:string"/>

</xs:choice>

</xs:complexType>

</xs:element>

Элемент-составитель all определяет неупорядоченный набор элементов, но в данном контексте не был рассмотрен.

3.2.5 ОГРАНИЧЕНИЯ

XML Schema также поддерживает ограничения уникальности и ключевые ограничения, которые обеспечивают целостность данных при обмене информацией.

Ограничения уникальности определяются с помощью элемента unique, который указывает элементы или атрибуты, которые должны быть уникальными. Например, для определения уникальности по фамилии и дате рождения работника:

<xs:unique>

<xs:selector xpath="."/>

<xs:field xpath="Фамилия"/>

<xs:field xpath="датаРождения"/>

</xs:unique>

Это ограничение будет применяться только в контексте элемента <СТУДЕНТ>.

Ограничения по ключу аналогичны ограничениям уникальности, но значение не должно быть пустым. Они моделируют первичный ключ реляционной базы данных:

<xs:key name="pkСтудент">

<xs:selector xpath="."/>

<xs:field xpath="Фамилия"/>

<xs:field xpath="датаРождения"/>

</xs:key>

Ссылочные ограничения позволяют ограничивать ссылки значениями указанных ключей. Например, если необходимо связать объект с <СТУДЕНТ>, можно определить ссылку так:

<xs:keyref refer="pkСтудент" name="fkСтудент">

<xs:selector xpath="."/>

<xs:field xpath="Фамилия"/>

<xs:field xpath="датаРождения"/>

</xs:keyref>

3.3 СТИЛИ И ФОРМАТИРОВАНИЕ ДАННЫХ XML

Создатели языка XML отделили содержимое документа от его структуры и форматирования. Важно понимать, что информация в XML-документе не определяет, как она будет отображена на экране. Для этого необходима дополнительная информация — таблица стилей, которая придаёт документу желаемый внешний вид. Данная концепция важна, так как в ER-модели информация о формате не учитывается, и её недостаток часто необходимо компенсировать при разработке экранных форм.

Таблица стилей — это специальный документ, содержащий список стилей, применяемых к информации XML-документа. Процесс описания того, как визуально представить содержимое документа, называется стилизацией.

Таблица стилей переводит информацию XML-документа в форму, подходящую для визуального отображения. Например, при просмотре документа в веб-браузере таблица стилей должна создать подходящий HTML-документ.

Технология таблиц стилей, которая преобразует документы в другие форматы, представлена спецификацией XSL. Основная идея состоит в том, что при трансформировании документа можно добавить информацию для визуализации и открытия в определенной программе, например, в веб-браузере.

Пример преобразования XML-документа

Рассмотрим пример XML-документа, содержащего данные о CD:

<?xml version="1.0" encoding="UTF-8"?>

<?xml-stylesheet type="text/xsl" href="catalog.xslt"?>

<catalog>

<cd>

<title>Empire Burlesque</title>

<artist>Bob Dylan</artist>

</cd>

<cd>

<title>Live a Paris</title>

<artist>Celin Dion</artist>

</cd>

</catalog>

Этот документ имеет древовидную структуру, где корневым элементом является <catalog>. Важно отметить, что корневой элемент определяет стартовую точку для XSL-процессора — приложения, обрабатывающего таблицу стилей и выполняющего трансформацию данных, например, в HTML-документ.

XSL-процессор ищет шаблоны, описывающие последовательности XML-документа. Когда он находит совпадение с именем элемента, он применяет соответствующий шаблон для трансформации данных.

Процесс преобразования:

1. Сравнение корневого узла с шаблонами:

Корневой узел XML-документа сопоставляется со всеми шаблонами в таблице стилей. Находится шаблон с атрибутом match="/", который указывает, что для корневого элемента будет сгенерирован HTML-код. Например, это может быть <html><body>.

1. Обработка дочерних узлов:

В элементе шаблона используется команда <xsl:apply-templates>, которая указывает процессору на необходимость обработать дочерние узлы корневого узла.

1. Проверка второго дочернего узла:

Корневой узел может иметь различные дочерние узлы, такие как <?xml-stylesheet?> и корневой элемент, в данном случае catalog. Для корневого элемента не определены правила, и процесс продолжается.

1. Обработка первого дочернего элемента:

Первая ветвь документа, которая будет обработана, — это элемент cd, представляющий альбом.

1. Сопоставление с шаблоном для элемента cd:

Для элемента cd находится шаблон <xsl:template match="cd">, который указывает, что необходимо вывести <h4>CD</h4>, а затем перейти к обработке названия альбома.

1. Извлечение названия альбома:

В соответствии с шаблоном, используется <xsl:apply-templates select="title">, что приводит к извлечению значения элемента title, например, «Empire Burlesque».

1. Добавление переноса строки:

После вывода названия добавляется перенос строки <br/>.

1. Извлечение имени исполнителя:

Используется <xsl:apply-templates select="artist">, что приводит к извлечению значения элемента artist, например, «Bob Dylan».

1. Обработка второго дочернего элемента:

Процесс повторяется для второго дочернего элемента, также cd, для альбома «Live a Paris».

1. Завершение обхода дерева:

После завершения обхода всех дочерних элементов возвращаемся к корневому шаблону, где закрываются теги </body></html>.

В результате преобразования данных из XML в HTML, сгенерированный HTML-документ будет готов к отображению в веб-браузере. При открытии XML-документа в браузере его таблица стилей автоматически обрабатывается, что позволяет пользователю увидеть форматированный результат, как показано на рисунке.