Основы использования XML в базах данных

Графеева Н.Г.

2017

Введение

В современных СУБД наряду с традиционным (реляционным) подходом к хранению и SQL-ной манерой манипуляции над данными широко используется представление данных в виде XML-структур и использование специализированных языков (XPath, XQuery) для манипуляций над такими данными. Настоящая презентация посвящена изучению этого вопроса.

Пример (xml-документ)

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
    <to> Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
</note>
```

XML - история, причины возникновения

- *XML* (*Extensible Markup Language)* – язык, ориентированный на разметку документов. Разметка текста осуществляется при помощи обрамляющих тегов. Создаваемые документы состоят из элементов (тегов) и текста, причем элементы помогают правильно понимать документ при чтении и обрабатывать его в электронном виде. Чем больше описательных элементов, тем больше частей документа можно идентифицировать.
- Языки разметки прошли путь от первых форм, создававшихся компаниями и госучреждениями, до Стандартного языка обобщенной разметки (Standard Generalized Markup Language - SGML), Гипертекстового языка разметки (Hypertext Markup Language - HTML) и в конечном итоге до XML. SGML может показаться сложным, а HTML (который, по сути, сначала был просто набором элементов) оказался недостаточно мощным для идентификации информации. XML разрабатывался как простой в применении и удобный для расширения язык разметки.
- В XML можно создавать свои собственные элементы, что позволяет точно представлять фрагменты данных. Документы можно не просто разделять на абзацы и заголовки, но и выделять любые фрагменты внутри документа. Чтобы это было эффективно, нужно определить конечный перечень своих элементов и придерживаться его. Элементы можно определять в Описании типа документа (Document Type Definition - DTD) или в схеме (XML Schema - xsd) , что будет кратко обсуждено далее.

Элементы XML

- Документы XML состоят из текста и разметки. Большая часть текста помещается в элементы, в которых текст окружен тегами. Например:

<recipename> *Ice Cream Sundae* </recipename>

- Теги образуют *элемент*, в который можно вводить текст и другие элементы (атрибуты).
- Имена элементов можно создавать как для отдельных документов, так и для групп документов. Можно указывать правила, которые должны соблюдаться для элементов.
- XML-документ может содержать пустые теги, внутри которых ничего нет и которые могут выражаться одним тегом, а не парой из открывающего и замыкающего тегов. Например, это может быть самостоятельный тег в стиле HTML:

<img src="mylogo.gif">

Декларация XML

- Первой строкой документа XML может быть декларация XML. Эта необязательная часть документа определяет его как документ XML, что может помочь автоматическим инструментам и людям распознавать документ как XML, а не как документ с другим способом разметки.
- Декларация может выглядеть просто как <?xml> или включать версию XML (<?xml version="1.0">) и даже кодировку символов.

Пример (декларация):

```
<?xml version="1.0" encoding="utf-8"?>
```

Корневой элемент

Начальный и замыкающий теги корневого элемента окружают весь текст XML-документа (за исключением декларации). В XML-документе должен присутствовать только один корневой элемент, и это необходимая "обложка" для него.

Пример (декларация + корневой элемент):

```
<?xml version="1.0" encoding="UTF-8"?>
<recipe>
        …….
</recipe>
```

Наименования элементов

- Имена элементов (тэги) могут содержать буквы, цифры и специальные знаки, такие как знак подчеркивания (_).
- Пробелы в именах элементов не допускаются.
- Имена должны начинаться с буквы, а не с цифры или знака.
- Регистр не имеет значения (за исключением первого и последнего тега), но во избежание путаницы соблюдайте его.

Пример:

```
<?xml version="1.0" encoding="UTF-8"?>
<recipe>
   <recipename>Ice Cream Sundae</recipename>
   <preptime>5 minutes</preptime>
</recipe>
```

Вложение элементов

- В XML-документах допустимо вложение элементов.
- ***Вложение***— это размещение элементов внутри других элементов. Эти новые элементы называются ***дочерними*** элементами, а элементы, которые их окружают, — их ***родительскими*** элементами.
- Вложение может делать XML-документ многоуровневым.
- Типичная синтаксическая ошибка связана с вложенностью родительского и дочернего элементов. Каждый дочерний элемент должен быть целиком расположен между открывающим и замыкающим тегами своего родительского элемента. Дочерние элементы должны заканчиваться до начала следующего дочернего элемента.

Пример (правильное вложение элементов)

```
<?xml version="1.0" encoding="UTF-8"?>
<recipe>
    <recipename>Ice Cream Sundae</recipename>
    <ingredlist>
         <listitem>
                <quantity>3</quantity>
                <itemdescription>
                        chocolate syrup or chocolate fudge
                </itemdescription>
        </listitem>
        <listitem>
                 <quantity>1</quantity>
                 <itemdescription>
                        nuts
                 </itemdescription>
        </listitem>
        <listitem>
                 <quantity>1</quantity>
                 <itemdescription>
                        cherry
                 </itemdescription>
        </listitem>
    </ingredlist>
   <preptime>5 minutes</preptime>
</recipe>
```

Атрибуты

- К элементам иногда добавляются ***атрибуты***. Атрибуты состоят из пары имя-значение, где значение берется в двойные кавычки ("), вот так: type="dessert". Атрибуты позволяют сохранять вместе с элементом дополнительные параметры, меняя значения этих параметров от элемента к элементу в одном и том же документе.
- Атрибут (или даже несколько атрибутов) указывается внутри начального тега элемента:

  <recipe type="dessert">

- При добавлении нескольких атрибутов они разделяются пробелами:

  <recipename cuisine="american" servings="1">

- Можно использовать любое количество атрибутов. Атрибуты особенно полезны, если документы будут храниться, например, по типу рецептов. Имена атрибутов могут содержать такие же символы, что и имена элементов, с теми же правилами исключения пробелов и начала имени с буквы.

Комментарии

- В XML-документ можно добавлять комментарии. Синтаксис:

<!-- Комментарий здесь -->

Пример (XML-документ с атрибутами и комментариями)

```
<?xml version="1.0" encoding="UTF-8"?>
<recipe type="dessert">
  <!– здесь имя рецепта -->
  <recipename cuisine="american" servings="1“>
        Ice Cream Sundae
  </recipename>
  <!– а здесь время приготовления -->
  <preptime>
        5 minutes
  </preptime>
</recipe>
```

Правильно и неправильно построенный XML-документ

- *Правильный XML*— это код XML, составленный с соблюдением всех правил XML: правильное именование элементов, вложение, именование атрибутов и т.п.
- *Под проверкой (validation)* понимается проверка структуры документа на соответствие установленным для нее правилам и определению дочерних элементов для каждого родительского элемента. Эти правила могут быть определены в *Описании типа документа* (dtd-файл) или в *Описании XML схемы* (xsd -файл) . Для такой проверки требуется создать dtd –файл или xsd-файл, а затем дать ссылку на соответствующий файл в XML-файле.

- Чтобы разрешить проверку, нужно ближе к началу своих XML-документов поместить декларацию типа документа (DOCTYPE). Эта строка содержит ссылку на dtd или xsd-файл , который будет использоваться для проверки данного документа. Строка DOCTYPE может быть примерно такой:

- <!DOCTYPE MyDocs SYSTEM "filename.dtd">

- <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" " http://www.w3.org/TR/html4/strict.dtd">

Пример (dtd-описание)

<!ELEMENT people_list (person*)>

<!ELEMENT person (name, birthdate?, gender?, socialsecuritynumber?)>
    <!ELEMENT name (#PCDATA) >

<!ELEMENT birthdate (#PCDATA) >

<!ELEMENT gender (#PCDATA) >

<!ELEMENT socialsecuritynumber (#PCDATA) >

Пример (использование dtd-описания)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE people_list SYSTEM "example.dtd">
<people_list>
   <person>
      <name>Fred Bloggs</name>
      <birthdate>27/11/2008</birthdate>
      <gender>Male</gender>

    <socialsecuritynumber>1234567890</socialsecuritynumber>
   </person>
</people_list>
```

Пример (xsd-описание)

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="country">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="country_name" type="xs:string"/>
            <xs:element name="population" type="xs:decimal"/>
         </xs:sequence>
         </xs:complexType>
   </xs:element>
</xs:schema>
```

Пример (xml, соответствующий xsd-описанию)

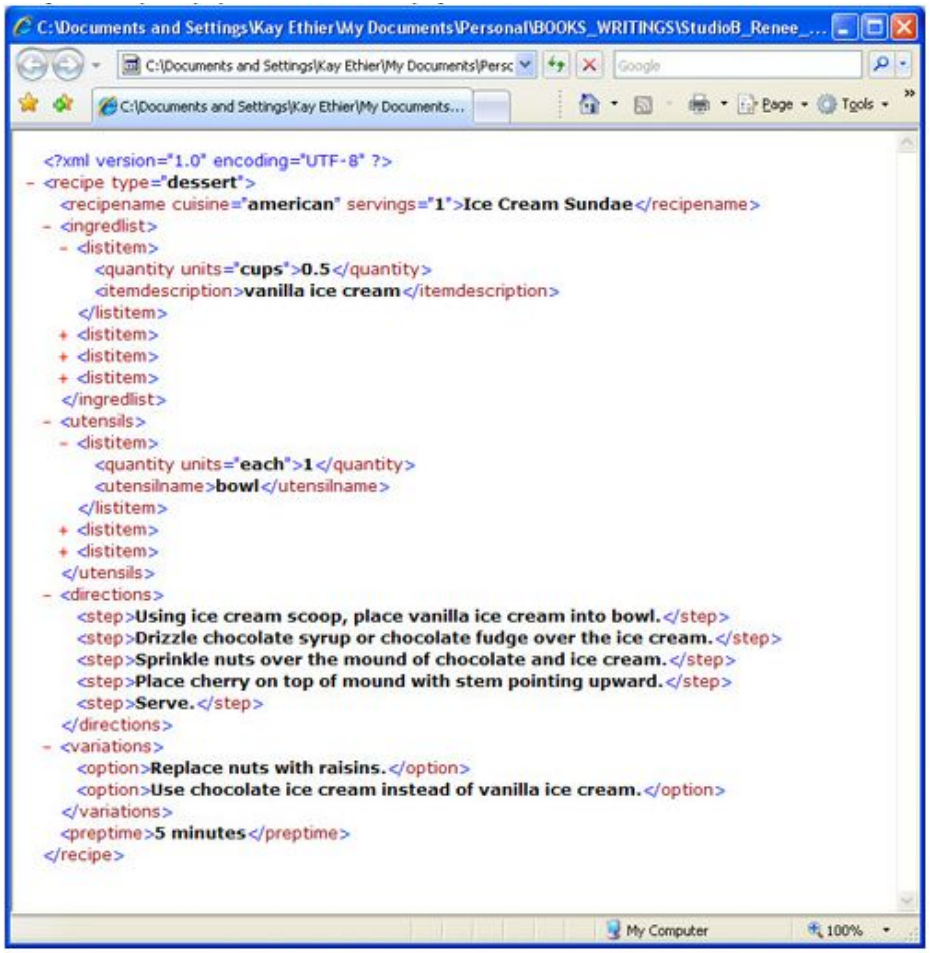
```
<?xml version="1.0" encoding="utf-8"?>
<country>
   <country_name>France</country_name>
   <population>59.7</population>
</country>
```

Проверка XML

- Самый простой способ – открыть документ в каком-нибудь редакторе XML или Internet Browser.

- *Примечание: это всего лишь проверка вложенных структур.*

Пример (как выглядит XML-документ в Internet Browser)



Проверка структуры xml-документа

В Internet существуют разнообразные <xml-валидаторы>, позволяющие проверить структуру xml-документа на соответствие его описанию (dtd или xsd) . Например, по адресу:

http://www.freeformatter.com/xml-validator-xsd.html

Formatters
» JSON Formatter
» HTML Formatter
» XML Formatter
» SQL Formatter

Validators
» JSON Validator
» HTML Validator
» **XML Validator - XSD**
» XSD Generator
» XPath Tester
» Credit Card Number Generator & Validator
» Regular Expression Tester

Encoders & Decoders
» Url Encoder & Decoder
» Base 64 Encoder & Decoder
» QR Code Generator

Code Minifiers
» JavaScript Minifier
» CSS Minifier

Converters
» XSLT (XSL Transformer)
» XML to JSON Converter
» JSON to XML Converter
» CSV to XML Converter
» Epoch Timestamp To Date

Cryptography
» Message Digester
» HMAC Generator

String Escaper & Utilities
» String Utilities
» HTML Escape

XML Validator - XSD (XML Schema)

Validates the XML string/file against the specified XSD string/file. XSD files are "XML Schemas" that describe the structure of a XML document. The validator checks for well formedness first, meaning that your XML file must be parsable using a DOM/SAX parser, and only then does it validate your XML against the XML Schema. The validator will report fatal errors, non-fatal errors and warnings.

There is no limit to the file you can upload but be patient with big or huge files.

XML Input

Option 1: Copy-paste your XML string here

Option 2: Or type in the URL to your XML file

http://www.example.com/myfile.xml

XSD Input

Option 1: Copy-paste your XSD string here

Option 2: Or type in the URL to your XSD file

http://www.example.com/myfile.xsd

VALIDATE XML

Как сохранить XML-документ в базе данных?

- В базах данных существуют специальные типы данных, предназначенные для хранения xml-документов:

- ORACLE - XMLType
- DB2 - XML

- Кроме того, в репозиторий **базы** можно загрузить dtd или xsd – описания загружаемых документов (чтобы потом проверять корректность загружаемых xml-документов)

Пример (ORACLE)

Загрузка xml-документа в базу ORACLE.

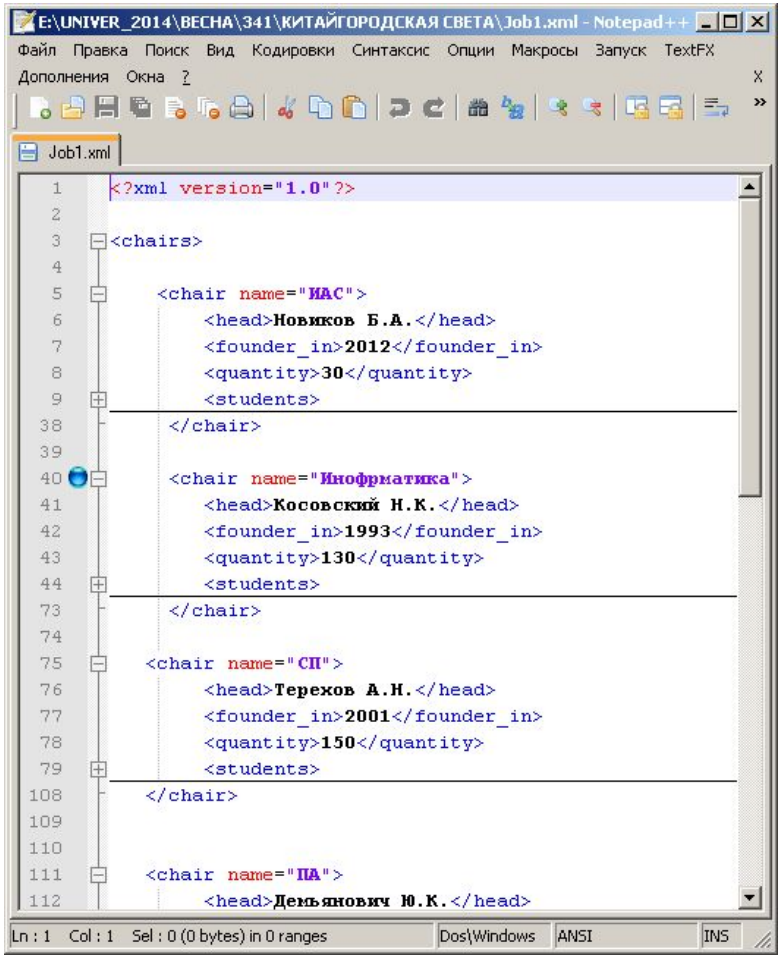
- 1.Создаем подходящую таблицу:

```
CREATE TABLE EMPLOYEES
(
  id     NUMBER,
  data   XMLTYPE
);
```

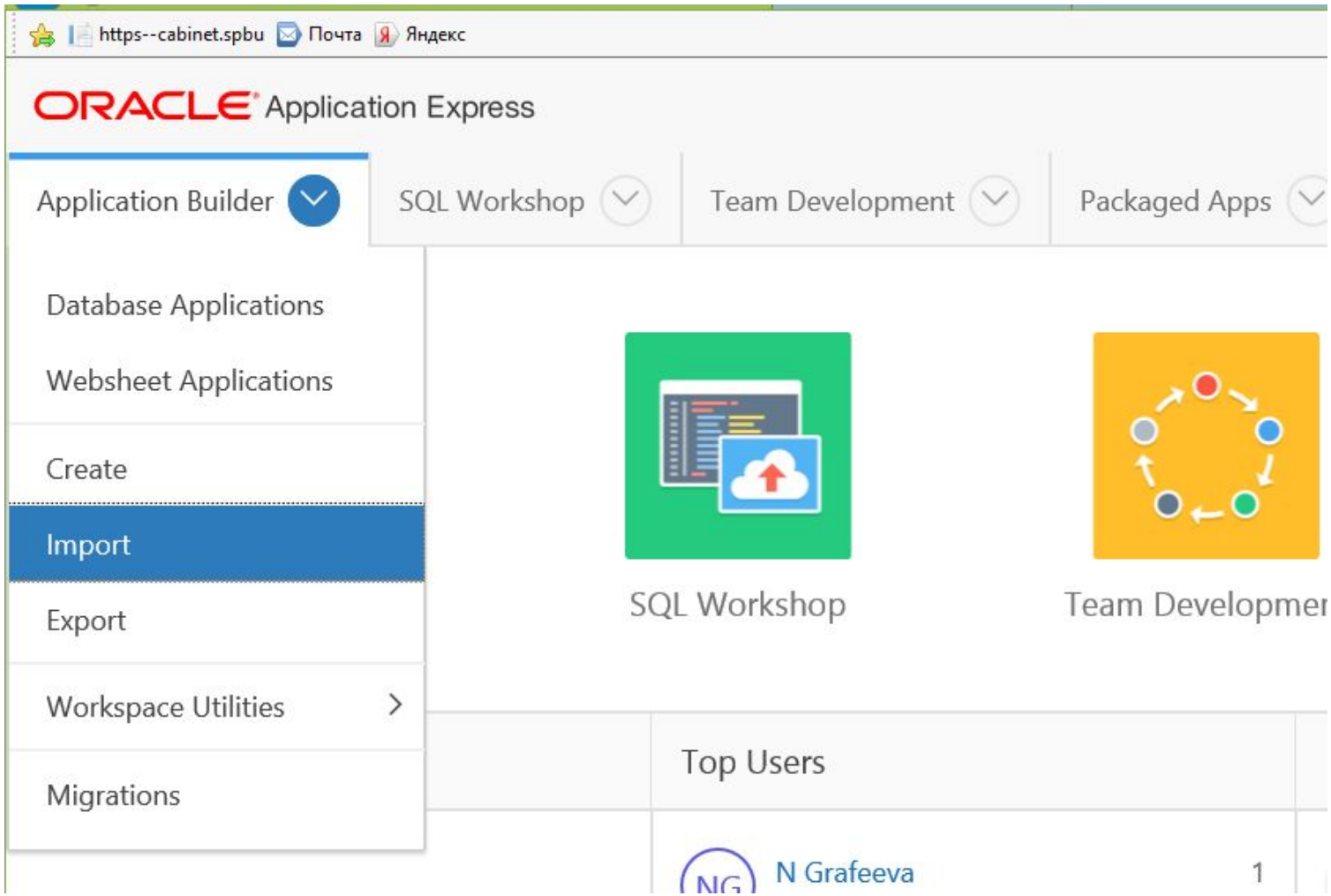
- 2. Загружаем **небольшой** xml-документ в подготовленную таблицу:

```
INSERT INTO EMPLOYEES
    VALUES (1, xmltype ('<Employees>
   <Employee emplid="1111" type="admin">
      <firstname>John</firstname>
      <lastname>Watson</lastname>
      <age>30</age>
      <email>johnwatson@sh.com</email>
   </Employee>
   <Employee emplid="2222" type="admin">
      <firstname>Sherlock</firstname>
      <lastname>Homes</lastname>
      <age>32</age>
      <email>sherlock@sh.com</email>
   </Employee>
   <Employee emplid="3333" type="user">
      <firstname>Jim</firstname>
      <lastname>Moriarty</lastname>
      <age>52</age>
      <email>jim@sh.com</email>
   </Employee>
   <Employee emplid="4444" type="user">
      <firstname>Mycroft</firstname>
      <lastname>Holmes</lastname>
      <age>41</age>
      <email>mycroft@sh.com</email>
   </Employee>
</Employees>'));
```

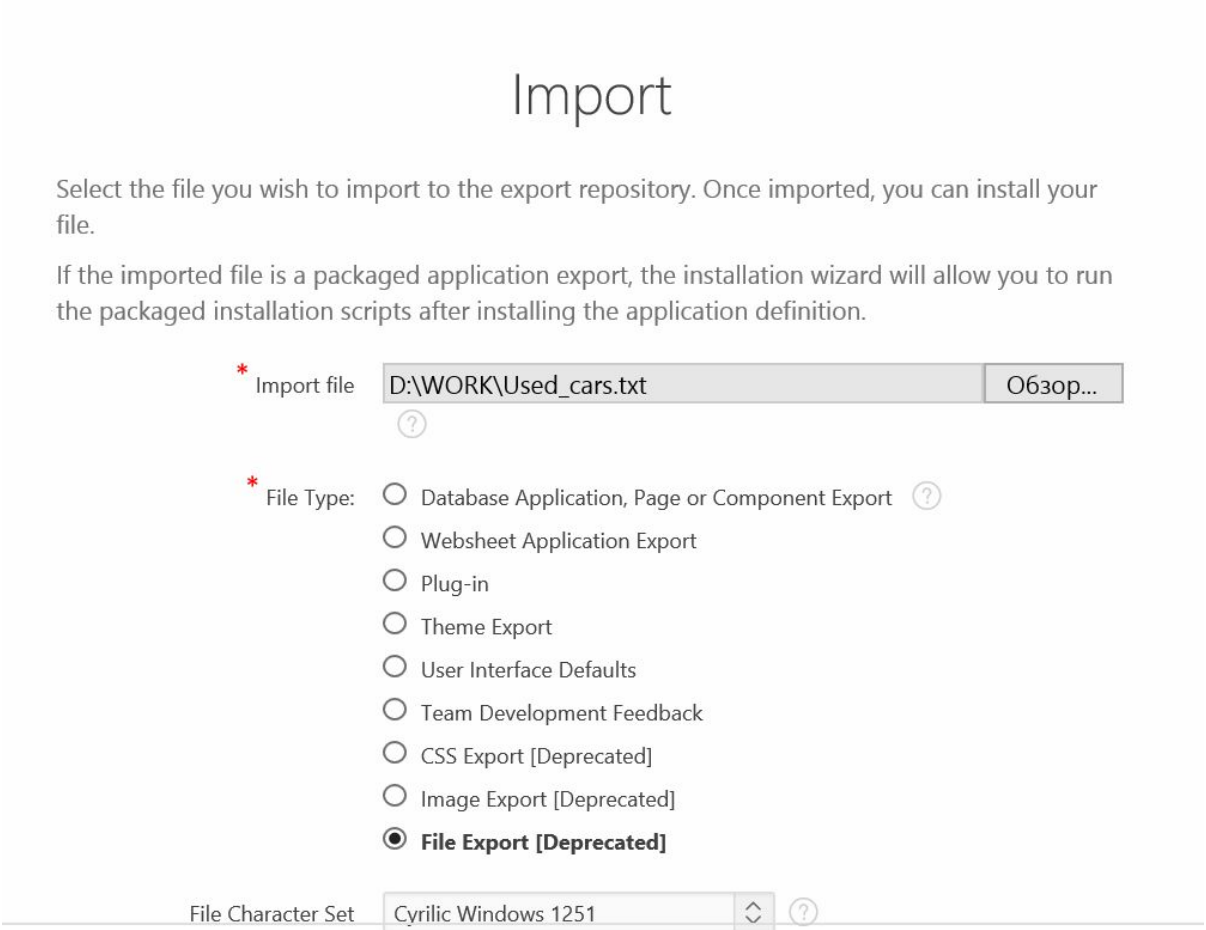
Как загрузить большой XML-документ через ORACLE APEX?

- 

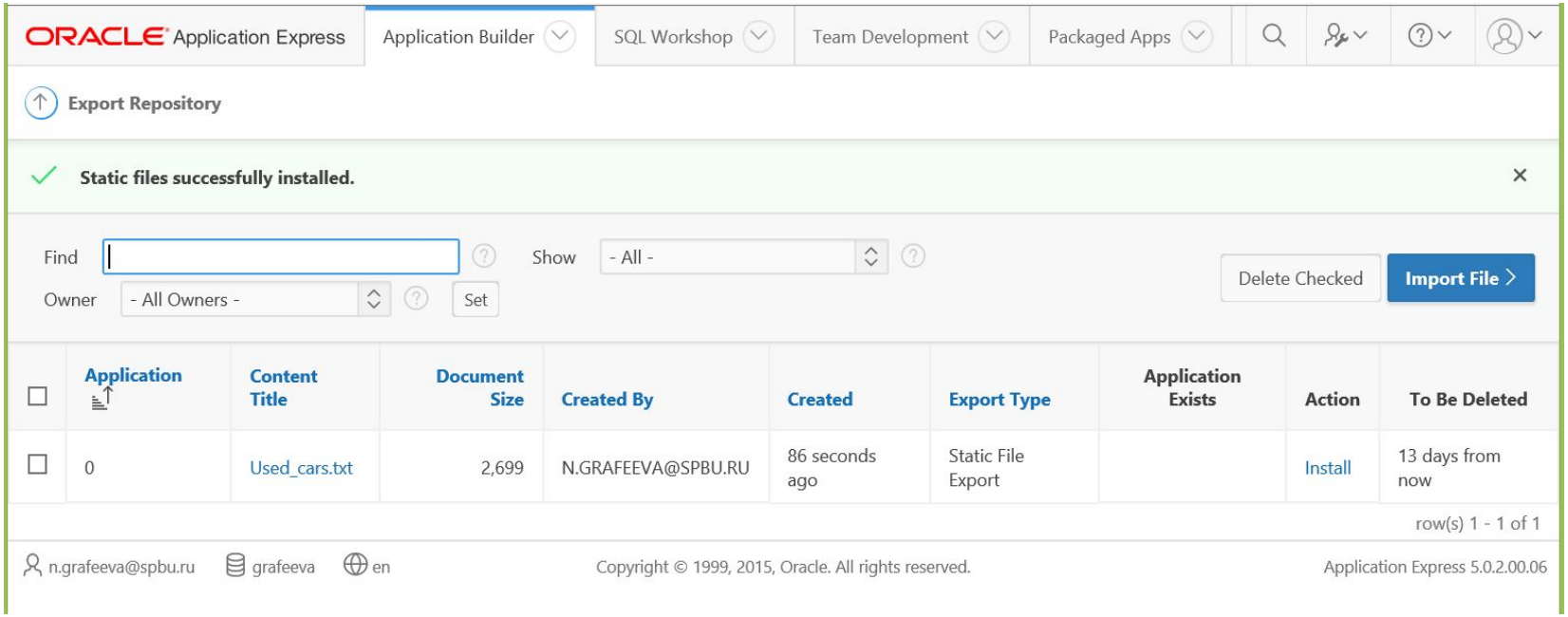
Application Builder -> Import

- 

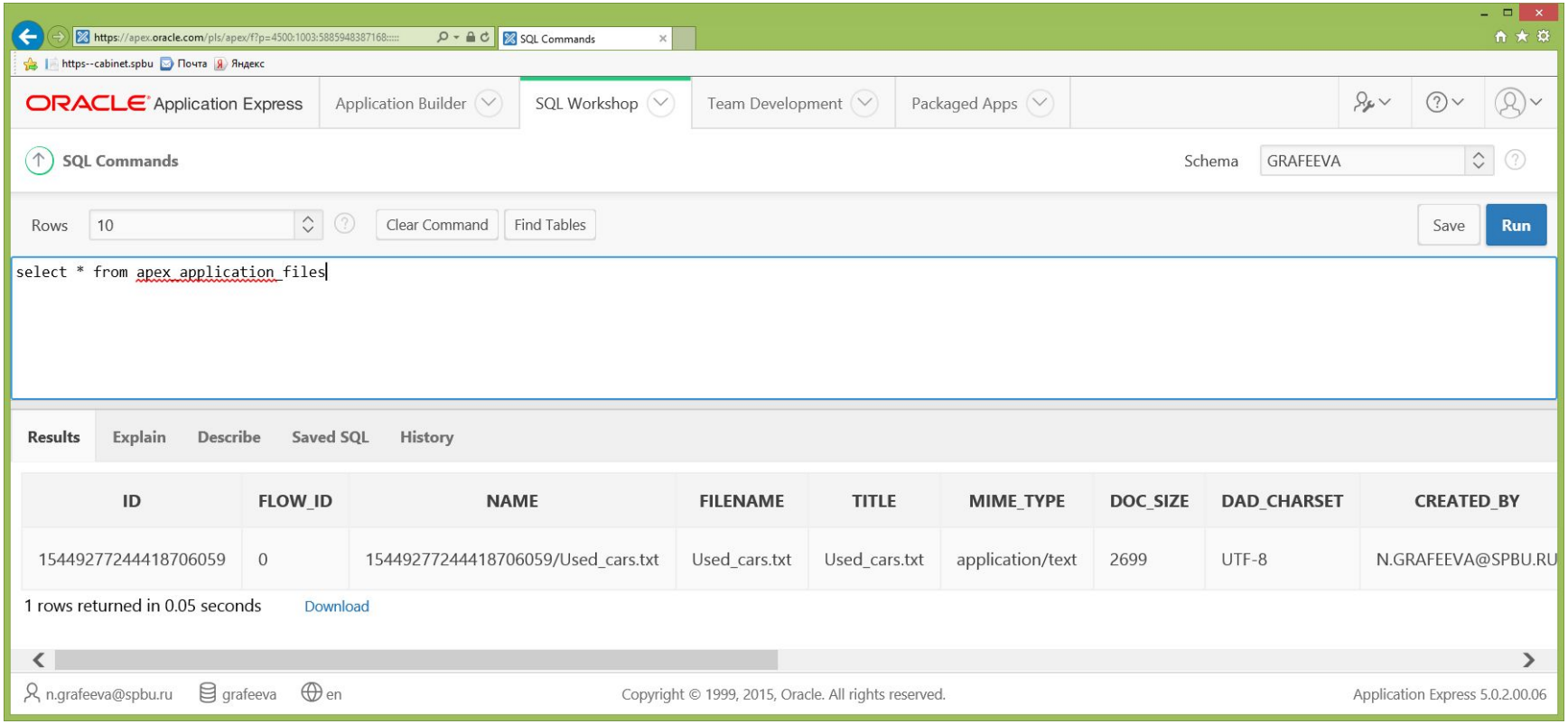
Укажем имя файла и кодировку



Проверим его наличие в репозитории рабочего пространства (Application Builder –> Repository)



Найдем файл через системное представление (apex_application_files)



Загрузка и преобразование файла типа BLOB в таблицу с полем типа XMLType

```
insert into employees(id, data)
select 3, xmltype(blob_content, 171)
/* 171 – соответствует кодировке win1251 */
from apex_application_files
where filename = ‘Used_cars.txt‘
```

Примечание: преобразование к типу XMLType нужно провести потому, что документ был загружен в поле типа BLOB.

Контрольная (5 баллов)

- Создать xml-файл (+ xsd или dtd описание) с данными об IT-кафедрах мат-меха (названия кафедр, заведующие кафедрами, студентами, имена, номерами зачеток, отметками, названиями предметов и т.п.). Проверить на соответствие в каком-нибудь инструменте.
- Загрузить xml-файл в специально подготовленную таблицу в схеме базы.

Полезные ссылки

- http://www.w3schools.com/xml