ORACLE XML DB

Some practical notes

Албена Захариева

ORACLE XML DB – дефиниция

Тази технология разширява релационната база данни, интегрирайки функционалност на native XML database.

ORACLE XML DB предоставя:

- нови методи за навигация и запитвания за XML document hierarchies.
- собствен XML data-type, който се използва за запазване и управление на XML documents.
- множество от методи и SQL оператори, които предоставят възможност за изпълнение на действия върху XML съдържанието.
- поддръжка на standard W3C XML Schema data model
- XML/SQL двойственост, допускайки XML операции върху SQL и SQL операции върху XML съдържание
- поддържа XPath

ORACLE XML DB предоставя:

- поддържа XML Repository, което допуска XML съдържанието да бъде организирано и управлявано чрез file / folder / URL
- APIs, чрез които се извършва достъп и редактиране на XML съдържание от Java, C и PL/SQL
- XML специфично управление на паметта и оптимизации

Подобрения в Oracle 10g

- По-висока производителност при записване и извършване на заявки към XML; оптимизация на използваната памет
- *еволюция на XML Schema:* по-голяма гъвкавост при смяна на XML Schema в зависимост на изискванията на приложението, като съществуващите документи и референции се запазват в зависимост от промените, които са направени върху XML схемата.

Подобрения в Oracle 10g

- NLS (National Language Support) – пълна поддръжка на хетерогенни character sets, заедно с protocol based access до Oracle XML DB repository, който предоставя възможност за трансформация в две посоки между client character set и database character set. Това означава, че XML DB repository може да бъде използвано за обмен на информация между клиенти използващи различни character sets.
- *обогатяване и развитие на използваните XPath изрази*
- нова функционалност, позволяваща обработката на много големи (Gigabyte) по размери XML документи

ORACLE XML DB предоставя:

- *XSLT Performance improvement:* оптимизация на XSLT engine, водеща до 120% подобрение в производителността (бързодействието) при трансформация на Schema-Based XML Type и 70% подобрение в трансформацията на Non-Schema-Based XML Type
- *Text Query Optimizations*: text searches се извършват използвайки ora:contains Xpath функцията, която се прилага използвайки Oracle Text index.
- *CTXXPATH Optimizations:* подобрения извършени в употребата на CTXXPATH индекса, позволявайки употребата на повече XPath изрази.

Основни характеристики

XML Type

- native server data-type, дефиниращ за базата данни, че колоната или таблицата съдържат XML

Основни характеристики

- предоставя методи за общи операции като schema validation и XSLT Transformation
- употреба като тип на PL/SQL променливи, както и при дефиниция и извикване на PL/SQL процедури и функции.
- обектен тип, даващ възможност за създаване на таблица от XMLType.
- допуска се употребата му при дефиниция на views – създаване на XMLType view или релационно view, което съдържа XMLType колона
- допуска съдържание само на валиден XML документ, чието съдържание се запазва като XML text, използвайки CLOB data type.

Основни характеристики

- съдържанието на таблицата или колоната може да бъде валидирано, съобразно XML Schema, което предоставя възможността за запазване на съдържанието на документа, чрез използване на structured-storage techniques, които водят до декомпозирането и запазването му като множество от SQL обекти, вместо като текст в CLOB. Обектният модел използван за запазване на документа автоматично се извлича от съдържанието на XML Schema.
- множество от методи, които предоставят възможността за извършване на основни операции като:

✓ извличане на подмножество от nodes, съдържащо се в XML Type - extract() , extractValue()

✓ проверка дали даден node съществува в XMLType – existsNode()

✓ валидиране на съдържанието на XMLType, чрез XML Schema - schemaValidate()

✓ прилагане на XSLT Transformation – transform()

Запазване на XML документи, чрез използване на :

- CLOB за XML документ, който не е базиран на XML Schema
- CLOB или множество от SQL обекти за Schema базиран XML документ
- Релационни Views, съдържащи колона от XML Type или XML Type views, за релационни или външни данни, които могат да бъдат представени като XML

Пример за създаване на таблица с XMLType колона, изпълнение на запис на ред в таблицата (Insert) и запитване (Query).

Създаване на таблица с XMLType колона:

```
CREATE TABLE  SFA_OBJECT(
        OBJECT_ID         NUMBER(16),
        OBJECT_CLASS  VARCHAR2(10),
       OBJECT_DATA   XMLTYPE
)
/
INSERT INTO SFA_OBJECT VALUES(
1000
,'Person'
,XMLTYPE(
      '<Object ClassID="Person">
        <PersonalData>
           <FullName>
                    <FirstName>Ivan</FirstName>
                    <MiddleName>Petrov</MiddleName>
                    <LastName>Petrov</LastName>
           </FullName>
```

```
 <BirthDate>
                <Day>13</Day>
                <Month>December</Month>
                <Year>1900</Year>
</BirthDate>
 <Addresses>
        <Address Type="Home">
          <Street>Dondukov</Street>
          <StreetNo>123</StreetNo>
          <Town>Sofia</Town>
          <PostCode>1504</PostCode>
          <Country Code="BG">Bulgaria</Country>
        </Address>
        <Address Type="Business">
                <Street>Vitoshka</Street>
                <StreetNo>12</StreetNo>
                <Town>Sofia</Town>
                <PostCode>1604</PostCode>
                <Country Code="BG">Bulgaria</Country>
        </Address>
```

```
<Address Type="Delivery">
        <Street>Vitoshka</Street>
        <StreetNo>126</StreetNo>
        <Town>Sofia</Town>
        <PostCode>1604</PostCode>
        <Country Code="BG">Bulgaria</Country>
</Address>
</Addresses>
 <Telephones>
      <Telephone Type="Home">
       <CountryCode>00359</CountryCode>
       <TelNumber>9898989</TelNumber>
     </Telephone>
```

```
<Telephone Type="Mobile">
        <CountryCode>00359</CountryCode>
     <TelNumber>0887565656</TelNumber>
</Telephone>
 <Telephone Type="Business">
            <CountryCode>0031</CountryCode>
            <TelNumber>0234020623</TelNumber>
            <Extension>10</Extension>
</Telephone>
</Telephones>
</PersonalData>
        <BusinessData>
            <CompanyName>VirtualConnection </CompanyName>
            <Department>Universal</Department>
            <Title>Representative</Title>
        </BusinessData>
</Object>'
))
/
```

Извличане на данни:

```
SELECT extractValue(X.OBJECT_DATA,
/Object[@ClassID=”Person”]/PersonalData/FullName/FirstName‘)
PersonName
FROM SFA_OBJECT X
WHERE existsNode(X.OBJECT_DATA,’/Object[BirthDate/Day=”13”]’) = 1
        /
```

Резултат:

PersonName
Ivan

```
SELECT extractValue(X.OBJECT_DATA,’//FirstName‘)
FROM SFA_OBJECT X
WHERE existsNode(X.OBJECT_DATA,’/Object[BirthDate/Day=”13”]’) = 1
```

XML Schema -

- стандарт за дефиниране структурата, съдържанието и семантиката на нов клас XML документи, ето защо много често се използва термина “instance document” за XML документ, който има статус валидиран спрямо дадена XML Schema.
- поддържа употребата на обектно-ориентирани техники като наследяване, предоставяйки възможността за дефиниране на сложни обекти от основни data types, дефинирани от XML Schema language.
- дефинира механизъм за валидиране на instance documents с дадена XML Schema

Пример:

```
<?xml version="1.0" encoding="UTF-8"?>
  <Object ClassID="Person"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation=
  "C:\SFA_OBJECT.xsd">

    <PersonalData> .....</ PersonalData>
    <BusinessData> ……..</BusinessData>
 </Object>
```

XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified">
   <xs:element name="PostCode">
      <xs:simpleType>
         <xs:restriction base="xs:short"/>
      </xs:simpleType>
   </xs:element>
   <xs:element name="Street">
      <xs:simpleType>
         <xs:restriction base="xs:string"/>
      </xs:simpleType>
    </xs:element>
    <xs:element name="StreetNo">
      <xs:simpleType>
         <xs:restriction base="xs:byte"/>
         </xs:simpleType>
      </xs:element>
```

```
<xs:element name="Town" type="xs:string"/>
<xs:complexType name="AddressType">
    <xs:sequence>
        <xs:element ref="Street"/>
        <xs:element ref="StreetNo"/>
        <xs:element ref="Town"/>
        <xs:element ref="PostCode"/>
        <xs:element name="Country“    type="CountryType"/>
    </xs:sequence>
    <xs:attribute name="Type" use="required">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value="Business"/>
                <xs:enumeration value="Delivery"/>
                <xs:enumeration value="Home"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
</xs:complexType>
```

```
<xs:complexType name="AddressesType">
   <xs:sequence>
      <xs:element name="Address" type="AddressType"
maxOccurs="unbounded"/>
   </xs:sequence>
 </xs:complexType>
<xs:element name="Day" type="xs:byte"/>
<xs:element name="Month" type="xs:string"/>
<xs:element name="Year" type="xs:short"/>
 <xs:complexType name="BirthDateType">
    <xs:sequence>
      <xs:element ref="Day"/>
       <xs:element ref="Month"/>
       <xs:element ref="Year"/>
    </xs:sequence>
</xs:complexType>
```

```
<xs:complexType name="CountryType">
        <xs:simpleContent>
        <xs:extension base="xs:string">
        <xs:attribute name="Code" type="xs:string“
   use="required"/>
      </xs:extension>
      </xs:simpleContent>
      </xs:complexType>
      <xs:element name="FirstName" type="xs:string"/>
      <xs:element name="LastName" type="xs:string"/>
      <xs:element name="MiddleName" type="xs:string"/>
      <xs:complexType name="FullNameType">
         <xs:sequence>
             <xs:element ref="FirstName"/>
             <xs:element ref="MiddleName"/>
             <xs:element ref="LastName"/>
         </xs:sequence>
     </xs:complexType>
```

```
<xs:element name="Object">
<xs:complexType>
  <xs:sequence>
      <xs:element name="PersonalData" type="PersonalDataType"/>
      <xs:element name="BusinessData" type="BusinessDataType"/>
   </xs:sequence>
  <xs:attribute name="ClassID" type="xs:string" use="required"/>
</xs:complexType>
</xs:element>
<xs:complexType name="PersonalDataType">
  <xs:sequence>
          <xs:element name="FullName" type="FullNameType"/>
          <xs:element name="BirthDate" type="BirthDateType"/>
          <xs:element name="Addresses" type="AddressesType"/>
          <xs:element name="Telephones" type="TelephonesType"/>
  </xs:sequence>
 </xs:complexType>
 <xs:element name="CompanyName" type="xs:string"/>
 <xs:element name="Department"       type="xs:string"/>
 <xs:element name="Title“                  type="xs:string"/>
```

```
<xs:complexType name="BusinessDataType">
      <xs:sequence>
            <xs:element ref="CompanyName"/>
            <xs:element ref="Department"/>
            <xs:element ref="Title"/>
      </xs:sequence>
</xs:complexType>
<xs:element name="CountryCode">
     <xs:simpleType><xs:restriction base="xs:short"/> </xs:simpleType>
</xs:element>
  <xs:element name="Extension" type="xs:byte"/>
  <xs:element name="TelNumber">
        <xs:simpleType>
             <xs:restriction base="xs:int"/>
           </xs:simpleType>
  </xs:element>
 <xs:complexType name="TelephoneType">
     <xs:sequence>
            <xs:element ref="CountryCode"/>
            <xs:element ref="TelNumber"/>
            <xs:element ref="Extension" minOccurs="0"/>
     </xs:sequence>
```

```
<xs:attribute name="Type" use="required">
 <xs:simpleType>
     <xs:restriction base="xs:string"/>
 </xs:simpleType>
</xs:attribute>
</xs:complexType>

<xs:complexType name="TelephonesType">
    <xs:sequence>
       <xs:element name="Telephone" type="TelephoneType"
    maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
</xs:schema>
```

Namespaces:

- дефинира механизъм за различаване/разграничаване на едни и същи имена използвани при дефиниция на един обект (елементи, атрибути, complex type, simple type) в две различни XML Schema. XML Namespace е колекция от имена, идентифициращи се от URI reference, която се използва в XML документи

XML Schema & Namespaces

- targetNamespace атрибута се включва в дефиницията на “schema” елемента и се използва за дефиниране на Namespace, свързан с дадена XML Schema. Ако XML Schema включва

XML Schema & Namespaces

targetNamespace всички елементи и типове, дефинирани от схемата са свързани с този *namespace*. Ако XML Schema не включва *targetNamespace* всички елементи и типове, дефинирани от схемата са свързани с *NULL namespace.*

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" декларация на XMLSchema-instance namespace в *root element* на instance document

- *noNamespaceSchemaLocation (Schema Location)* атрибут идентифициращ, че XML Schema няма определена *namespace* декларация за обектите, които дефинира; съдържанието на

XML Schema & Namespaces

noNamespaceSchemaLocation атрибута обикновено е в формата на URL, описващ къде се намира прилаганата/използваната XMLSchema.

Пример:

Namespaces

<Object ClassID="Person"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:\SFA_OBJECT.xsd"> -
тази XML Schema дефинира глобален елемент Object, не съдържа targetNamespace декларация

xsi:noNamespaceSchemaLocation=http://xmlns.oracle.com/demo/sfaObject.xsd

SQL / XML

SQL / XML операторите попадат в две категории:

- първата категория се състои от множество оператори, които правят възможно query и достъп на XML съдържание като част от нормални SQL операции.
- втората категория се състои от множество оператори, които предоставят възможност за генериране на XML от резултат, получен от SQL Select statement
  - ✓ **extract()** - изпълнява XPath expression и връща като резултат node или nodes, които отговарят на поставеното условие като XML документ или фрагмент

SQL/XML

- **extractValue()** - изпълнява XPath expression и връща резултат в подходящ SQL data type
- **updatexml()** - извършва частичен update на XML документ, отговарящ на множество от XPath expressions. Всеки от XPath expression идентифицира target node в документа и новата стойност за този node.
- **xmlsequence()** - превръща XML документ в множество от добре форматирани XML документи

SQL/XML дефинира следните елементи:

- ***XMLAgg*:** функция за групиране или събиране на данни в XML
- ***XMLAttributes*:** функция за поставяне на атрибут в XML elements

SQL/XML

- ***XMLConcat***: функция за обединяване на две или повече XML стойности
- ***XMLElement***: функция за превръщане на релационна стойност в XML element, във формат: <elementName>value</elementName>; **return XMLType value**
- ***XMLForest***: функция за генериране на лист, наречен “гора” от XML elements от релационни стойности

Примери:

SQL/XML

```
SQL> describe COUNTY

 Name                      Type
 ------------              ------------
 COUNTY_NAME               VARCHAR2(10)
 STATE                     VARCHAR2(2)

SQL> describe ATTRACTION

 Name                      Type
 ------------              ------------
 COUNTY_NAME               VARCHAR2(10)
 ATTRACTION_NAME           VARCHAR2(30)
 ATTRACTION_URL            VARCHAR2(40)
 GOVERNMENT_OWNED          CHAR(1)
 LOCATION                  VARCHAR2(20)
```

SQL/XML

```
SELECT XMLElement("Attraction",attraction_name)
       AttractionXmlElement
FROM attraction;
```

```
AttractionXmlElement
----------------------------------------
<Attraction>Disney Land</Attraction>
<Attraction>Sofia Land</Attraction>
```

Nested XMLElements:

```
SELECT XMLElement("Attraction",
      XMLAttributes(government_owned AS GOV),
                  XMLElement("Name",attraction_name),
            XMLElement("Location",location),
            XMLElement("URL",attraction_url))
      XmlElementWithAttr
FROM attraction;
```

XmlElementWithAttr

```
<Attraction GOV="Y">
  <Name>Disney Land</Name>
  <Location>Paris</Location>
  <URL>http://www.nps.gov/piro/</URL>
</Attraction>
```

Null Values

```
SELECT XMLElement("Attraction",
            XMLAttributes(government_owned AS     GOV),
            XMLForest(attraction_name AS "Name",
                        Location AS "Location",
                        attraction_url AS "URL")) Attraction
FROM attraction
WHERE attraction_name='Sofia Land';
```

Резултат:

```
Attraction
----------------------------------------------
<Attraction GOV="Y">
  <Name>Sofia Land</Name>
  <URL>http://www.dir.bg/sofia-land.html</URL>
</Attraction>
```

XMLAgg е aggregate function, като MIN, MAX и AVG.

```
SELECT XMLElement("County",
          XMLAttributes(c.county_name AS "Name",
       'http://www.w3.org/2001/XMLSchema-instance'
                          AS "xmlns:xsi",
      'http://gennick.com/tourist.xsd'
                         AS "xsi:noNamespaceSchemaLocation"),
          XMLAgg(
              XMLElement("Attraction",
              XMLAttributes(government_owned AS GOV),
             XMLForest(a.attraction_name AS "Name",
                        a.Location AS "Location",
                       a.attraction_url AS "URL"))))
FROM county c INNER JOIN attraction a
    ON c.county_name = a.county_name
GROUP BY c.country_name
```

Резултат:

```
<Country Name=”Bulgaria” xmlns:xsi=
      “http://www.w3.org/2001/XMLSchema-instance”
      xsi:noNamespaceSchemaLocation =
      “http://gennick.com/tourist.xsd”>
<Attraction GOV="Y">
  <Name>Sofia Land</Name>
  <Location>Sofia</Location>
  <URL>http://www.dir.bg/sofia-land.html</URL>
</Attraction>
<Attraction GOV="N">
  <Name>Patilanci</Name>
  <Location>Sofia</Location>
  <URL>http://www.dir.bg/patilanci-land.html</URL>
</Attraction>
</Country>
```

Създаване на XMLType view за представяне на XML data as files, в директория на XML DB Repository.

При отварянето на такъв файл, съдържанието на файла се конструира “on the fly” чрез изпълнението на query.

```
CREATE OR REPLACE VIEW ATTRACTION_XML
  OF XMLType
  WITH OBJECT ID ( /* генерира unique object identifier за
      всеки ред във view */
    extractValue(sys_nc_rowinfo$,'/County/@Name')
      /*reference to sys_nc_rowinfo$ е референция към
      текущия ред ( "current" row in the view)*/
   /* XPath query syntax '/County/@Name' връща стойност
      за Name attribute в всеки <County>      element, който
      след това се използва като основа за генериране на
      unique identifier за всеки ред, върнат в view.
  */
  ) from columns (c.County_name)
AS
```

```
SELECT XMLElement("County",
      XMLAttributes(c.county_name AS "Name",
                  'http://www.w3.org/2001/XMLSchema-instance'
                  AS "xmlns:xsi",
                  'http://gennick.com/tourist.xsd'
                    AS "xsi:noNamespaceSchemaLocation"),
XMLAgg(
                  XMLElement("Attraction",
                  XMLAttributes(government_owned AS GOV),
                  XMLForest(a.attraction_name AS "Name",
                  a.Location AS "Location",
                  a.attraction_url AS "URL")))) x
FROM county c INNER JOIN attraction a
    ON c.county_name = a.county_name
GROUP BY c.county_name;
```

За появата на XML документите в repository е необходимо създаването на repository resources

Първо е необходимо създаването на folder с име /ATTRACTIONS

Създаване на repository document за всеки XML document.

```
DECLARE
  CURSOR attractions_key IS
    SELECT DISTINCT county_name
    FROM attraction;
    /*retrieve all distinct county names from the attraction
     table on which the view is based. */
   xmlref REF XMLType;
   result BOOLEAN;
```

```
BEGIN
    FOR key_rec IN attractions_key LOOP
    SELECT MAKE_REF(attraction_xml, key_rec.county_name)
INTO xmlref
     FROM dual;

     /*MAKE_REF генерира a REF за всеки county's row in the
       view's result set.*/
     result := DBMS_XDB.createResource(
               '/ATTRACTIONS/' || key_rec.county_name || '.xml'
               ,xmlref);
     /*REF се използва в извикването на
       DBMS_XDB.createResource, която създава  resource в
       XML DB Repository  и свързва resource към данните за
       дадена страна от view's result set.  Resource се появява
       като .xml file в repository.
   */
     END LOOP;
END;
```

XSLT Transformations:

```
<!-- Example.xml -->

<ROWSET>
 <ROW num="1">
  <EMPNO>7839</EMPNO>
  <ENAME>KING</ENAME>
 </ROW>
 <ROW num="2">
  <EMPNO>7788</EMPNO>
  <ENAME>SCOTT</ENAME>
 </ROW>
</ROWSET>
```

```
<xsl:stylesheet version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
      <body>
        <xsl:for-each select="ROWSET">
          <table border="1" cellspacing="0">
            <xsl:for-each select="ROW">
              <tr>
                <td><xsl:value-of select=="num"/></td>
                <td><xsl:value-of select="EMPNO"/></td>
<td><xsl:value-of select="ENAME"/></td>
              </tr>
            </xsl:for-each>
          </table>
        </xsl:for-each>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

Резултат:

1	7839	King
2	7788	Scott