

RIGA TECHNICAL UNIVERSITY

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

INSTITUTE OF APPLIED COMPUTER SYSTEMS

“Technology of Large Databases”

Practical assignment #4

**Design of XML Database**

Author: Doston hamrakulov

Studentcardno.: 151ADB089

2017/ 2018 study year

Contents

[Assignment description: 1](#_Toc500648161)

[1. design and create XML documents and schema files 2](#_Toc500648162)

[1.1 XML document 2](#_Toc500648163)

[1.1.1 Owner.xml 2](#_Toc500648164)

[1.1.2 ItemStore.xml 3](#_Toc500648165)

[1.2 XSD Schema 4](#_Toc500648166)

[1.2.1 Owner.xsd 4](#_Toc500648167)

[1.2.2 ItemStore.xsd 5](#_Toc500648168)

[2. Creating oracle directory to store files 7](#_Toc500648169)

[Step\_#1 7](#_Toc500648170)

[Step\_#2 8](#_Toc500648171)

[Step\_#3 XML schema registration using BFILE: 9](#_Toc500648172)

[2. Unstructured type (CLOB type) 12](#_Toc500648173)

[Creation 12](#_Toc500648174)

[Data input 12](#_Toc500648175)

[Query\_#1: EXTRACT() 13](#_Toc500648176)

[Query\_#2: EXTRACT() 13](#_Toc500648177)

[Query\_#3: GETCLOBVAL() 14](#_Toc500648178)

[Query\_#4 EXTRACTVALUE() 14](#_Toc500648179)

[Query\_#5 EXTRACT from Owner 15](#_Toc500648180)

[Query\_#6 15](#_Toc500648181)

[3. Structured type (OR type); 15](#_Toc500648182)

[Creation 15](#_Toc500648183)

[Data input: 16](#_Toc500648184)

[Query\_#1: EXTRACT() 16](#_Toc500648185)

[Query\_#2: GETCLOBVAL() 17](#_Toc500648186)

[Query\_#3: EXTRACTVALUE() 17](#_Toc500648187)

[2.3 If You use Oracle12, binary type. 18](#_Toc500648188)

[4. Relational type of data from XML stored data; 18](#_Toc500648189)

[Creation 19](#_Toc500648190)

[Loading XML in Relation Table 19](#_Toc500648191)

[Data output 20](#_Toc500648192)

[5. XML type of data from relational type of data. 20](#_Toc500648193)

[Creation 20](#_Toc500648194)

[Data Input 21](#_Toc500648195)

[Generating XML data from Employee\_table 22](#_Toc500648196)

[6. Conclusions. 23](#_Toc500648197)

# Assignment description:

1. With XML editor (EditX, Oxygen, …) design and create XML documents and schema files.  
2. Design and creation of XML database data storage object-relational tables with XMLType use:  
– unstructured type (CLOB type);  
– structured type (OR type);  
– if You use Oracle12, binary type.  
3. Perform input of data (as text and use **bfile** (directory creation and use of large object function **bfile**)).  
4. Perform extraction of data (queries (4)) from tables.  
5. Perform registration of schema.  
6. Create table for structured type of storage with use of schema.  
7. Perform 4 queries.  
8. Perform queries:  
– to achieve relational type of data from XML stored data;  
– to achieve XML type of data from relational type of data.  
9. Conclusions.

# design and create XML documents and schema files

Text bold and italic SQL code

First of all, I am going to create 3 xml files and then I will create schema

files regarding those 3 created xml files.

## 1.1 XML document

### 1.1.1 Owner.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> <OwnerStore>  <Owner id="110200">  <Name>Doston</Name>  <Surname>Hamrakulov</Surname>  <Contact>  <Email>doston.hamrakulov@gmail.com</Email>  <Phone>28786188</Phone>  </Contact>  <Address>  <Street>Elizabetes 103/9</Street>  <City>Riga</City>  <ZipCode>LV-1050</ZipCode>  <Country>Latvia</Country>  </Address>  </Owner>  <Owner id="110201">  <Name>Janis</Name>  <Surname>Grabis</Surname>  <Contact>  <Email>janis@gmail.com</Email>  <Phone>99999999</Phone>  </Contact>  <Address>  <Street>Setas 1</Street>  <City>Riga</City>  <ZipCode>LV-1048</ZipCode>  <Country>Latvia</Country>  </Address>  </Owner>  <Owner id="110203">  <Name>Kate</Name>  <Surname>Adams</Surname>  <Contact>  <Email>kate.adams@mail.com</Email>  <Phone>9898089089</Phone>  </Contact>  <Address>  <Street>Hill str 10</Street>  <City>London</City>  <ZipCode>001000</ZipCode>  <Country>United Kingdom</Country>  </Address>  </Owner> </OwnerStore> |

### 1.1.2 ItemStore.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> <ItemStore>  <Item id="1">  <ItemType type="phone">  <PhoneType model="apple">  <Marka>Iphone 7s</Marka>  <Memory>32</Memory>  <Description>Отличное состояние, носился в чехле. Только телефон</Description>  <Price>500</Price>  </PhoneType>  </ItemType>  </Item>  <Item id="2">  <ItemType type="laptop">  <PhoneType model="lenovo">  <Marka>Ideapad 500</Marka>  <Memory>500</Memory>  <Description>Piedāvājam mazlietotu portatīvo datoru Lenovo Ideapad 500,   Procesors(CPU): Intel Core i7-6500U 2 kodoli 4 tredi ar 2.50Ghz frekvenci.  Operētājsistēma(OS): Microsoft Windows 10</Description>  <Price>500</Price>  </PhoneType>  </ItemType>  </Item>  <Item id="2">  <ItemType type="tv">  <PhoneType model="samsung">  <Marka>samsung lcd 777</Marka>  <Memory>100</Memory>  <Description>Lcd televizors 82 см, idealā stavoklī. Pults. Iespeja stiprinat pie sienas, Datoru pieslegšana, txt. Garantija.   Hd Ready. 16/9 Zoom. USB  TV tuner:Analog + Dekoders DVBT  </Description>  <Price>500</Price>  </PhoneType>  </ItemType>  </Item>  <Item id="4">  <ItemType type="phone">  <PhoneType model="samsung">  <Marka>Note 8</Marka>  <Memory>100</Memory>  <Description>Apskatīt un iegādāties preci var "Vita Lombards" filiālē Rīga, Merķeļa iela 7  Strādājam bez brīvdienām no 08:00 lidz 20:00.   Pārdošanā ir arī citi modeļi, un citas preces.   Ja nepieciešams vairāk informācijas, zvaniet uz norādīto telefona numuru.  </Description>  <Price>500</Price>  </PhoneType>  </ItemType>  </Item> </ItemStore> |

## 1.2 XSD Schema

### 1.2.1 Owner.xsd

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">  <xs:element name="OwnerStore">  <xs:complexType>  <xs:sequence>  <xs:element maxOccurs="unbounded" ref="Owner"/>  </xs:sequence>  </xs:complexType>  </xs:element>  <xs:element name="Owner">  <xs:complexType>  <xs:sequence>  <xs:element ref="Name"/>  <xs:element ref="Surname"/>  <xs:element ref="Contact"/>  <xs:element ref="Address"/>  </xs:sequence>  <xs:attribute name="id" use="required" type="xs:integer"/>  </xs:complexType>  </xs:element>  <xs:element name="Name" type="xs:NCName"/>  <xs:element name="Surname" type="xs:NCName"/>  <xs:element name="Contact">  <xs:complexType>  <xs:sequence>  <xs:element ref="Email"/>  <xs:element ref="Phone"/>  </xs:sequence>  </xs:complexType>  </xs:element>  <xs:element name="Email" type="xs:string"/>  <xs:element name="Phone" type="xs:integer"/>  <xs:element name="Address">  <xs:complexType>  <xs:sequence>  <xs:element ref="Street"/>  <xs:element ref="City"/>  <xs:element ref="ZipCode"/>  <xs:element ref="Country"/>  </xs:sequence>  </xs:complexType>  </xs:element>  <xs:element name="Street" type="xs:string"/>  <xs:element name="City" type="xs:NCName"/>  <xs:element name="ZipCode" type="xs:NMTOKEN"/>  <xs:element name="Country" type="xs:string"/> </xs:schema> |
| Graphical representation: |

### 1.2.2 ItemStore.xsd

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">  <xs:element name="ItemStore">  <xs:complexType>  <xs:sequence>  <xs:element maxOccurs="unbounded" ref="Item"/>  </xs:sequence>  </xs:complexType>  </xs:element>  <xs:element name="Item">  <xs:complexType>  <xs:sequence>  <xs:element ref="ItemType"/>  </xs:sequence>  <xs:attribute name="id" use="required" type="xs:integer"/>  </xs:complexType>  </xs:element>  <xs:element name="ItemType">  <xs:complexType>  <xs:sequence>  <xs:element ref="PhoneType"/>  </xs:sequence>  <xs:attribute name="type" use="required" type="xs:NCName"/>  </xs:complexType>  </xs:element>  <xs:element name="PhoneType">  <xs:complexType>  <xs:sequence>  <xs:element ref="Marka"/>  <xs:element ref="Memory"/>  <xs:element ref="Description"/>  <xs:element ref="Price"/>  </xs:sequence>  <xs:attribute name="model" use="required" type="xs:NCName"/>  </xs:complexType>  </xs:element>  <xs:element name="Marka" type="xs:string"/>  <xs:element name="Memory" type="xs:integer"/>  <xs:element name="Description" type="xs:string"/>  <xs:element name="Price" type="xs:integer"/> </xs:schema> |
| Graphical representation: |

# 2. Creating oracle directory to store files

In previous section, I have shown the creation of xml document and xml schema. Then I saved them in “E:\RTU lessons\Databases and object technology\Practical work\_3” and now I am going to load those files into my database.

I will show two different way to upload files into database:

## Step\_#1

|  |
| --- |
| C:/>sqlplus / as sysdba  sql>alter session set container=pdborcl;  SQL>GRANT CREATE any directory, DROP any directory, CONNECT, resource, unlimited tablespace, ALTER SESSION, CREATE VIEW TO HR IDENTIFIED BY HR;  Grant succeeded.  SQL>ALTER USER HR DEFAULT tablespace USERS TEMPORARY tablespace TEMP;  User altered.  SQL>CONNECT HR/HR@PDBORCL  connected  SQL>CREATE OR REPLACE directory XMLFILES AS 'E:\RTU lessons\Databases and object technology\Practical work\_3';  Directory created  SQL> |
| In CMD:    In SQL Developer |

## Step\_#2

|  |
| --- |
| 1. Directory of XML files definition:  create or replace directory XMLDIR as 'E:/RTU lessons/Databases and object technology/Practical work\_3/XMLSchema'; |
| In SQL Developer |

## Step\_#3 XML schema registration using BFILE:

Here I am going to register XML schema which I have created in Oxygen and uploaded from ‘E:\RTU lessons\Databases and object technology\Practical work\_3\XMLSchema’ in previous section.

|  |
| --- |
| XML Schema ItemStore.xsd  <?xml version="1.0" encoding="UTF-8"?> <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">  <xs:element name="ItemStore">  <xs:complexType>  <xs:sequence>  <xs:element maxOccurs="unbounded" ref="Item"/>  </xs:sequence>  </xs:complexType>  </xs:element>  <xs:element name="Item">  <xs:complexType>  <xs:sequence>  <xs:element ref="ItemType"/>  </xs:sequence>  <xs:attribute name="id" use="required" type="xs:integer"/>  </xs:complexType>  </xs:element>  <xs:element name="ItemType">  <xs:complexType>  <xs:sequence>  <xs:element ref="PhoneType"/>  </xs:sequence>  <xs:attribute name="type" use="required" type="xs:NCName"/>  </xs:complexType>  </xs:element>  <xs:element name="PhoneType">  <xs:complexType>  <xs:sequence>  <xs:element ref="Marka"/>  <xs:element ref="Memory"/>  <xs:element ref="Description"/>  <xs:element ref="Price"/>  </xs:sequence>  <xs:attribute name="model" use="required" type="xs:NCName"/>  </xs:complexType>  </xs:element>  <xs:element name="Marka" type="xs:string"/>  <xs:element name="Memory" type="xs:integer"/>  <xs:element name="Description" type="xs:string"/>  <xs:element name="Price" type="xs:integer"/> </xs:schema> |
| Registration  begin  DBMS\_XMLSCHEMA.REGISTERSCHEMA(  SCHEMAURL => 'http://localhost:8080/public/ItemStore.xsd',  SCHEMADOC => bfilename('XMLDIR', 'ItemStore.xsd'),  LOCAL => TRUE,  CSID => nls\_charset\_id('AL32UTF8') ) ;  end; |
| Result : |

Now I am going to register another **Owner.xsd** XML schema and then I will implement CLOB with that schema:

|  |
| --- |
| XML Schema ItemStore.xsd  <?xml version="1.0" encoding="UTF-8"?> <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">  <xs:element name="OwnerStore">  <xs:complexType>  <xs:sequence>  <xs:element maxOccurs="unbounded" ref="Owner"/>  </xs:sequence>  </xs:complexType>  </xs:element>  <xs:element name="Owner">  <xs:complexType>  <xs:sequence>  <xs:element ref="Name"/>  <xs:element ref="Surname"/>  <xs:element ref="Contact"/>  <xs:element ref="Address"/>  </xs:sequence>  <xs:attribute name="id" use="required" type="xs:integer"/>  </xs:complexType>  </xs:element>  <xs:element name="Name" type="xs:NCName"/>  <xs:element name="Surname" type="xs:NCName"/>  <xs:element name="Contact">  <xs:complexType>  <xs:sequence>  <xs:element ref="Email"/>  <xs:element ref="Phone"/>  </xs:sequence>  </xs:complexType>  </xs:element>  <xs:element name="Email" type="xs:string"/>  <xs:element name="Phone" type="xs:integer"/>  <xs:element name="Address">  <xs:complexType>  <xs:sequence>  <xs:element ref="Street"/>  <xs:element ref="City"/>  <xs:element ref="ZipCode"/>  <xs:element ref="Country"/>  </xs:sequence>  </xs:complexType>  </xs:element>  <xs:element name="Street" type="xs:string"/>  <xs:element name="City" type="xs:NCName"/>  <xs:element name="ZipCode" type="xs:NMTOKEN"/>  <xs:element name="Country" type="xs:string"/> </xs:schema> |
| Registration  begin  DBMS\_XMLSCHEMA.REGISTERSCHEMA(  SCHEMAURL => 'http://localhost:8080/public/Owner.xsd',  SCHEMADOC => bfilename('XMLDIR', 'Owner.xsd'),  LOCAL => TRUE,  CSID => nls\_charset\_id('AL32UTF8') ) ;  end; |
| Result : |

# Unstructured type (CLOB type)

## Creation

|  |
| --- |
| Item\_store table creation:  create table Item\_store of XMLType  XMLTYPE store as CLOB;  / |
| Result: |

|  |
| --- |
| Owner table creation:  create table Owner of XMLType  XMLTYPE STORE as CLOB  XMLSCHEMA  "http://localhost:8080/public/Owner.xsd"  ELEMENT "OwnerStore"; |
| Result: |

## Data input

|  |
| --- |
| From Istem\_store.xml to Item\_store table  insert into Item\_store values (xmltype(  bfilename('XMLDIR', 'ItemStore.xml'),  nls\_charset\_id('AL32UTF8') ) );  / |
| Result: |
| From Owner.xml to Owner table  insert into Owner values (xmltype(  bfilename('XMLDIR', 'Owner.xml'),  nls\_charset\_id('AL32UTF8') ) );  / |
| Result: |

## Query\_#1: EXTRACT()

|  |
| --- |
| Query:  select EXTRACT(OBJECT\_VALUE, '/ItemStore/Item[1]') "Item"  from Item\_store;  / |
| Result: |

## Query\_#2: EXTRACT()

|  |
| --- |
| Query:  SELECT EXTRACT(OBJECT\_VALUE, '/ItemStore/Item[4]/ItemType/PhoneType/Marka') as Phone\_marka,  EXTRACT(OBJECT\_VALUE, '/ItemStore/Item[4]/ItemType/PhoneType/Memory') as Memory,  EXTRACT(OBJECT\_VALUE, '/ItemStore/Item[4]/ItemType/PhoneType/Description/text()') as Description  FROM Item\_store;  / |
| Result: |

## Query\_#3: GETCLOBVAL()

|  |
| --- |
| Query:  select EXTRACT(OBJECT\_VALUE, '/ItemStore/Item[1]/ItemType/PhoneType/Marka') "Item"  from Item\_store;  / |
| Result: |

## Query\_#4 EXTRACTVALUE()

|  |
| --- |
| Query:  SELECT EXTRACTVALUE(OBJECT\_VALUE, '/ItemStore/Item[4]/ItemType/PhoneType/Marka') as Phone\_marka,  EXTRACTVALUE(OBJECT\_VALUE, '/ItemStore/Item[4]/ItemType/PhoneType/Memory') as Memory,  EXTRACTVALUE(OBJECT\_VALUE, '/ItemStore/Item[4]/ItemType/PhoneType/Description') as Description  FROM Item\_store;  / |
| Result: |

## Query\_#5 EXTRACT from Owner

|  |
| --- |
| Data output:  select EXTRACT(OBJECT\_VALUE, '/OwnerStore/Owner[1]/Name') "Owner Name"  from Owner; |
| Result: |

## Query\_#6

|  |
| --- |
| Data output:  select EXTRACT(OBJECT\_VALUE, '//OwnerStore/Owner[1]/Contact/Email/text()') "Owner Name"  from Owner; |
| Result: |

# Structured type (OR type);

Here, I am going to implement XMLType data structured storage

## Creation

|  |
| --- |
| create table Owner\_s of XMLType  XMLTYPE store AS OBJECT RELATIONAL  XMLSCHEMA "http://localhost:8080/public/Owner.xsd"  ELEMENT "OwnerStore"; |
| Result: |

## Data input:

|  |
| --- |
| insert into Owner\_s values (xmltype(  bfilename('XMLDIR', 'Owner.xml'),  nls\_charset\_id('AL32UTF8') ) );  / |
| Result: |

## Query\_#1: EXTRACT()

|  |
| --- |
| **SELECT EXTRACT(OBJECT\_VALUE, '/OwnerStore/Owner[1]/Name/text()') as Name\_owner,**  **EXTRACT(OBJECT\_VALUE, '/OwnerStore/Owner[1]/Contact/Email/text()') as Email\_owner,**  **EXTRACT(OBJECT\_VALUE, '/OwnerStore/Owner[1]/Address/Street/text()') as Address,**  **EXTRACT(OBJECT\_VALUE, '/OwnerStore/Owner[1]/Address/City/text()') as City**  **FROM Owner\_s;** |
| Result: |

## Query\_#2: GETCLOBVAL()

|  |
| --- |
| Query:  select A.OBJECT\_VALUE.GETCLOBVAL()  FROM Owner\_s A; |
| Result: |

## Query\_#3: EXTRACTVALUE()

|  |
| --- |
| Query:  SELECT EXTRACTVALUE(OBJECT\_VALUE, '/OwnerStore/Owner[3]/Name') as Name,  EXTRACTVALUE(OBJECT\_VALUE, '/OwnerStore/Owner[3]/Address/Street') as Address,  EXTRACTVALUE(OBJECT\_VALUE, '/OwnerStore/Owner[3]/Address/City') as Country,  EXTRACTVALUE(OBJECT\_VALUE, '/OwnerStore/Owner[3]/Contact/Email') as Email  FROM Owner\_s; |
| Result: |

## 2.3 If You use Oracle12, binary type.

I am using Oracle 11 g version. I wish I had Oracle 12 database to implement binary type and also to learn how JSON storage works in Oracle, because this format is very popular in modern Web.

# Relational type of data from XML stored data;

I have created new xml file for this task which is Student.xml and move it new location:

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> <Students>  <Student>  <name>Doston</name>  <surname>Hamrakulov</surname>  <email>doston.hamrakulov@gmail.com</email>  <degree>Bachelor</degree>  <mark>10</mark>  <country>Uzbekistan</country>  </Student>  <Student>  <name>Bobosher</name>  <surname>Hamrakulov</surname>  <email>bobosher.hamrakulov@gmail.com</email>  <degree>Master</degree>  <mark>10</mark>  <country>Uzbekistan</country>  </Student>  <Student>  <name>Kate</name>  <surname>Chaien</surname>  <email>Kate@gmail.com</email>  <degree>Master</degree>  <mark>8</mark>  <country>United Kingdom</country>  </Student>  <Student>  <name>Janis</name>  <surname>Grabis</surname>  <email>janis@gmail.com</email>  <degree>Bachelor</degree>  <mark>8</mark>  <country>Latvia</country>  </Student>  <Student>  <name>John</name>  <surname>Smith</surname>  <email>john.smith@gmail.com</email>  <degree>Master</degree>  <mark>7</mark>  <country>United States</country>  </Student> </Students> |

|  |
| --- |
| Creating new directory:  create or replace directory MY\_DIR as 'C:\XML\_files\_for database\Practical\_work'; |
| Result of execution: |

## Creation

|  |
| --- |
| create table tab\_students(  name varchar2(20),  surname varchar2(20),  email varchar2(20),  degree varchar2(20),  mark varchar2(20),  country varchar2(20)  );  / |
| Result of execution: |

## Loading XML in Relation Table

|  |
| --- |
| insert into tab\_students (name, surname, email, degree, mark, country)  select x.name, x.surname, x.email, x.degree, x.mark, x.country  from xmltable(  '/Students/Student'  passing xmltype(  bfilename('MY\_DIR', 'Student.xml'),  nls\_charset\_id('AL32UTF8')  )  columns id integer path '@cid',  name varchar2(20) path 'name',  surname varchar2(20) path 'surname',  email varchar2(20) path 'email',  degree varchar2(30) path 'degree',  mark varchar2(20) path 'mark',  country varchar2(20) path 'country'  ) x  ; |
| Result of execution: |

## Data output

|  |
| --- |
| Query:  Select \* from tab\_students; |
| Result of execution: |

# XML type of data from relational type of data.

## Creation

|  |
| --- |
| CREATE TABLE Employee\_table (  employee\_id NUMBER PRIMARY KEY,  first\_name VARCHAR2(30),  last\_name VARCHAR2(30),  email VARCHAR2(30),  hire\_date DATE,  job\_id varchar2(30),  salar NUMBER,  department VARCHAR2(30)  ); |
| Result of execution: |

## Data Input

|  |
| --- |
| INSERT INTO Employee\_table values(1, 'Doston', 'Hamrakulov', 'doston@gmail.com', '01-AUG-04', 'SA\_REP', 9000, 'DBP');  INSERT INTO Employee\_table values(2, 'Ever', 'Boston', 'ever@gmail.com', '09-SEP-12', 'SA\_REP', 7000, 'DSP');  INSERT INTO Employee\_table values(3, 'Adburahim', 'Salim', 'abdu2498r@gmail.com', '24-JAN-10', 'RTU\_ABD', 8000, 'DRP');  INSERT INTO Employee\_table values(4, 'Bek', 'Asim', 'BEK@gmail.com', '09-JUN-11', 'SA\_REP', 7000, 'DSP');  INSERT INTO Employee\_table values(5, 'Sher', 'Utkur', 'sher7709@gmail.com', '15-MAY-15', 'SA\_REP', 7700, 'DSP');  INSERT INTO Employee\_table values(6, 'Akosh', 'Rustamov', 'otash@gmail.com', '09-SEP-12', 'SA\_REP', 7000, 'DSP'); |
| Result of execution: |

## Generating XML data from Employee\_table

|  |
| --- |
| SELECT XMLElement("Employee", XMLAttributes(E.EMPLOYEE\_ID as "iD"),  XMLForest(E.FIRST\_NAME as "name", E.LAST\_NAME as "surname",  E.EMAIL as "email", E.HIRE\_DATE as "hire\_date", E.SALAR as "salary",  E.DEPARTMENT as "department")).EXTRACT('/\*') AS XML  FROM EMPLOYEE\_TABLE E; |
| Result of execution: |

As we expected, exactly 6 XMLs were generated because there are 6 rows in Employee\_table. To see the whole XML document, we could use **Single Record View** in **Query Result** tab:

|  |
| --- |
|  |

# Conclusions.

In this task, I have gained a lot of new knowledge about xml and how to work in database with xml. Moreover, I had opportunity to use different queries to retrieve data from XML and I have learned them too, such as XQuery.

In additional, I have acquired new skills how to work with Oxygen, before I was not aware of the cool software and it helped me more.