

RIGA TECHNICAL UNIVERSITY

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

INSTITUTE OF APPLIED COMPUTER SYSTEMS

“Technology of Large Databases”

Practical assignment #4

**XML Database**

Author: Ulugbek Akhmedjanov

Studentcardno.: 141ADB069

2016 / 2017study year

Contents:

[Assignment description 3](#_Toc469825859)

[Part #1: create XMLs and schema files(XSD) 4](#_Toc469825860)

[Part #2: create tables 9](#_Toc469825861)

[Part #3: input of data 11](#_Toc469825862)

[Part #4: Data extraction. 14](#_Toc469825863)

[Part #5: XMLQuery examples. 17](#_Toc469825864)

[Part #6: XML type of data from relational type data. 19](#_Toc469825865)

[Part #7: Relational data from XML. 21](#_Toc469825866)

[Conclusions: 23](#_Toc469825867)

# Assignment description

## Design of XML Database.

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);

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.

# Part #1: create XMLs and schema files(XSD)

Firstly, I am going to create a few XML samples and corresponding XSD schemas. To create XML documents and schema files, I used an application called **Oxygen XML**.

## XML document #1, Events.xml

This XML file will be used to demonstrate unstructuredXML storage with(see schema below) and without schema.

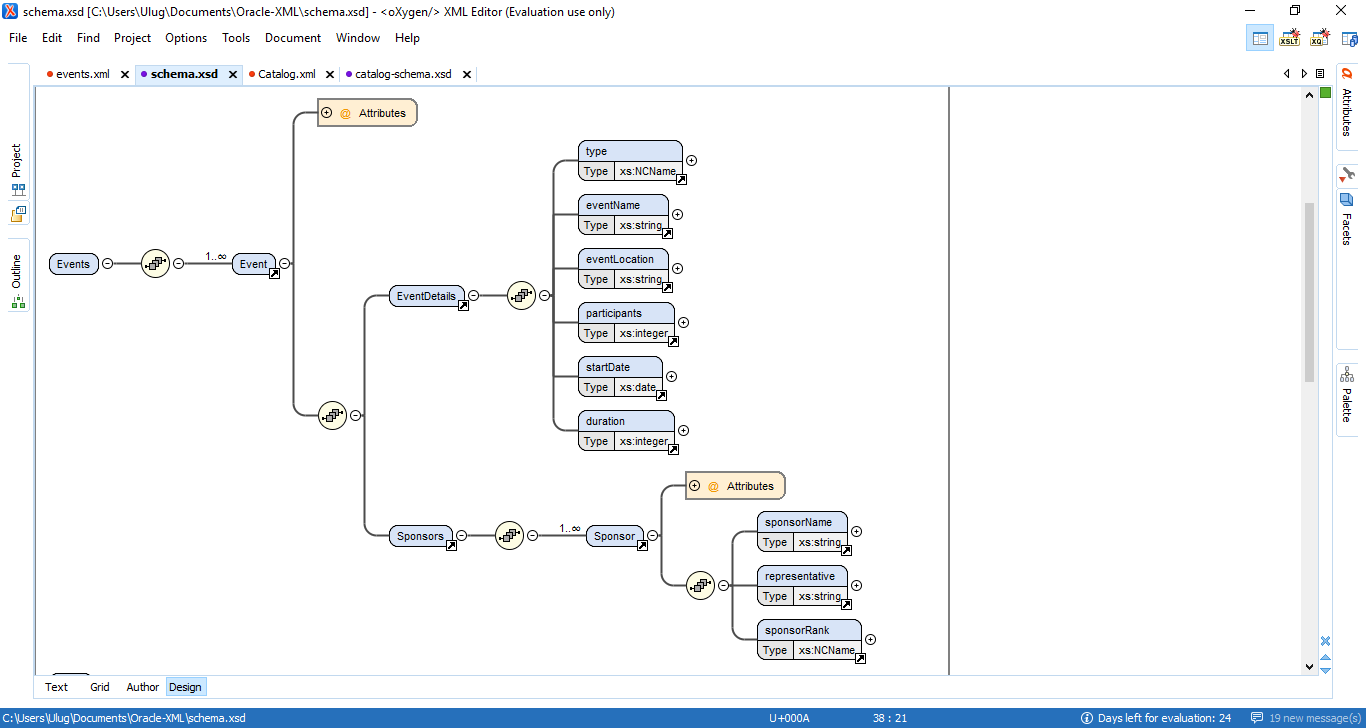
<?xmlversion="1.0" encoding="UTF-8"?>  
<Events>  
 <Eventid="1">  
 <EventDetails>  
 <type>Conference</type>  
 <eventName>Devoxx</eventName>  
 <eventLocation>Antwerpen, Belgium</eventLocation>  
 <participants>1000</participants>  
 <startDate>2016-12-11</startDate>  
 <duration>2</duration>  
 </EventDetails>  
 <Sponsors>  
 <Sponsorid="1">  
 <sponsorName>Oracle</sponsorName>  
 <representative>BrianGoetz</representative>  
 <sponsorRank>Golden</sponsorRank>  
 </Sponsor>  
 <Sponsorid="2">  
 <sponsorName>RedHat</sponsorName>  
 <representative>PeterDavis</representative>  
 <sponsorRank>Silver</sponsorRank>  
 </Sponsor>  
 </Sponsors>  
 </Event>  
 <Eventid="2">  
 <EventDetails>  
 <type>Meetup</type>  
 <eventName>Berlin JUG</eventName>  
 <eventLocation>Berlin, Germany</eventLocation>  
 <participants>55</participants>  
 <startDate>2016-06-10</startDate>  
 <duration>1</duration>  
 </EventDetails>  
 <Sponsors>  
 <Sponsorid="3">  
 <sponsorName>Siemens</sponsorName>  
 <representative>JuliusShaw</representative>  
 <sponsorRank>Platinum</sponsorRank>  
 </Sponsor>  
 </Sponsors>  
 </Event>  
</Events>

## XML document #2, Catalog.xml

<?xmlversion="1.0" encoding="UTF-8"?>  
<BookCatalog>  
 <Bookid="bk101">  
 <author>Gambardella, Matthew</author>  
 <title>XML Developer'sGuide</title>  
 <genre>Computer</genre>  
 <price>44.95</price>  
 <publish\_date>2000-10-01</publish\_date>  
 <description>Anin-depthlookatcreatingapplicationswith XML.</description>  
 </Book>  
 <Bookid="bk102">  
 <author>Ralls, Kim</author>  
 <title>MidnightRain</title>  
 <genre>Fantasy</genre>  
 <price>5.95</price>  
 <publish\_date>2000-12-16</publish\_date>  
 <description>A formerarchitectbattlescorporatezombies, anevilsorceress, andherownchildhoodtobecomequeenoftheworld.</description>  
 </Book>  
 <Bookid="bk103">  
 <author>Corets, Eva</author>  
 <title>MaeveAscendant</title>  
 <genre>Fantasy</genre>  
 <price>5.95</price>  
 <publish\_date>2000-11-17</publish\_date>  
 <description>Afterthecollapseof a nanotechnologysocietyinEngland, theyoungsurvivorslaythefoundationfor a newsociety.</description>  
 </Book>  
 <Bookid="bk104">  
 <author>Corets, Eva</author>  
 <title>Oberon'sLegacy</title>  
 <genre>Fantasy</genre>  
 <price>5.95</price>  
 <publish\_date>2001-03-10</publish\_date>  
 <description>Inpost-apocalypseEngland, themysteriousagentknownonlyasOberonhelpstocreate a newlifefortheinhabitantsofLondon. SequeltoMaeveAscendant.</description>  
 </Book>  
 <Bookid="bk105">  
 <author>Corets, Eva</author>  
 <title>TheSunderedGrail</title>  
 <genre>Fantasy</genre>  
 <price>5.95</price>  
 <publish\_date>2001-09-10</publish\_date>  
 <description>ThetwodaughtersofMaeve, half-sisters, battleoneanotherforcontrolofEngland. SequeltoOberon'sLegacy.</description>  
 </Book>  
 <Bookid="bk106">  
 <author>Randall, Cynthia</author>  
 <title>LoverBirds</title>  
 <genre>Romance</genre>  
 <price>4.95</price>  
 <publish\_date>2000-09-02</publish\_date>  
 <description>WhenCarlameetsPaulatanornithologyconference, tempersflyasfeathersgetruffled.</description>  
 </Book>  
</BookCatalog>

## XSD schema #1, events-schema.xsd

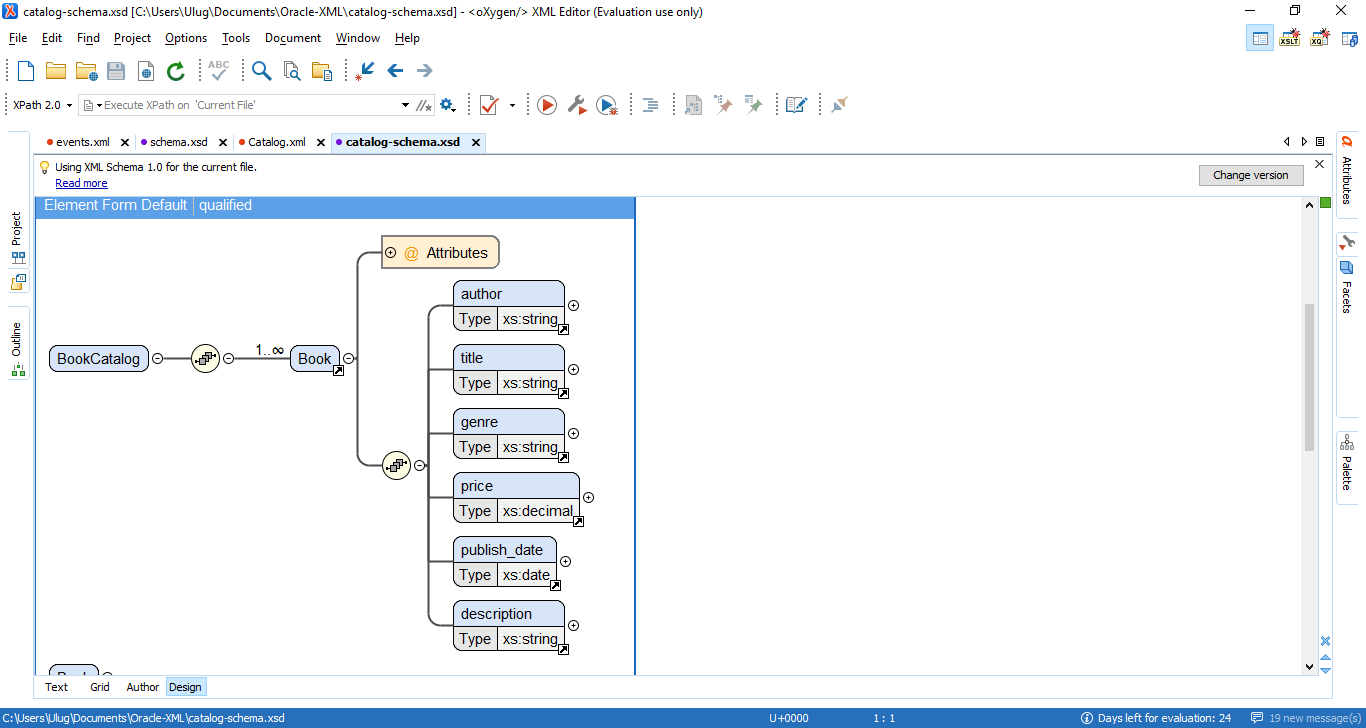
Here is the graphical representation and schema file content for the **Events.xml** document shown above.



<?xml version="1.0" encoding="UTF-8"?>  
<xs:schemaxmlns:xs="http://www.w3.org/2001/XMLSchema"elementFormDefault="qualified">  
<xs:element name="Events">  
<xs:complexType>  
<xs:sequence>  
<xs:elementmaxOccurs="unbounded" ref="Event"/>  
</xs:sequence>  
</xs:complexType>  
</xs:element>  
<xs:element name="Event">  
<xs:complexType>  
<xs:sequence>  
<xs:element ref="EventDetails"/>  
<xs:element ref="Sponsors"/>  
</xs:sequence>  
<xs:attribute name="id" use="required" type="xs:integer"/>  
</xs:complexType>  
</xs:element>  
<xs:element name="EventDetails">  
<xs:complexType>  
<xs:sequence>  
<xs:element ref="type"/>  
<xs:element ref="eventName"/>  
<xs:element ref="eventLocation"/>  
<xs:element ref="participants"/>  
<xs:element ref="startDate"/>  
<xs:element ref="duration"/>  
</xs:sequence>  
</xs:complexType>  
</xs:element>  
<xs:element name="type" type="xs:NCName"/>  
<xs:element name="eventName" type="xs:string"/>  
<xs:element name="eventLocation" type="xs:string"/>  
<xs:element name="participants" type="xs:integer"/>  
<xs:element name="startDate" type="xs:date"/>  
<xs:element name="duration" type="xs:integer"/>  
<xs:element name="Sponsors">  
<xs:complexType>  
<xs:sequence>  
<xs:elementmaxOccurs="unbounded" ref="Sponsor"/>  
</xs:sequence>  
</xs:complexType>  
</xs:element>  
<xs:element name="Sponsor">  
<xs:complexType>  
<xs:sequence>  
<xs:element ref="sponsorName"/>  
<xs:element ref="representative"/>  
<xs:element ref="sponsorRank"/>  
</xs:sequence>  
<xs:attribute name="id" use="required" type="xs:integer"/>  
</xs:complexType>  
</xs:element>  
<xs:element name="sponsorName" type="xs:string"/>  
<xs:element name="representative" type="xs:string"/>  
<xs:element name="sponsorRank" type="xs:NCName"/>  
</xs:schema>

## XSD schema #2, catalog-schema.xsd

This schema is based on **Catalog.xml**file and will be used for demonstration of structured storage type.

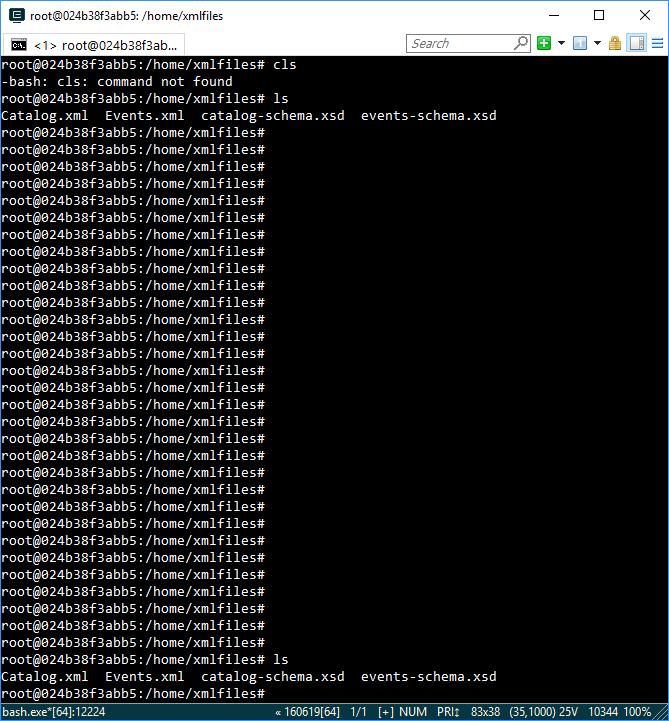


<?xml version="1.0" encoding="UTF-8"?>  
<xs:schemaxmlns:xs="http://www.w3.org/2001/XMLSchema"elementFormDefault="qualified">  
<xs:element name="BookCatalog">  
<xs:complexType>  
<xs:sequence>  
<xs:elementmaxOccurs="unbounded" ref="Book"/>  
</xs:sequence>  
</xs:complexType>  
</xs:element>  
<xs:element name="Book">  
<xs:complexType>  
<xs:sequence>  
<xs:element ref="author"/>  
<xs:element ref="title"/>  
<xs:element ref="genre"/>  
<xs:element ref="price"/>  
<xs:element ref="publish\_date"/>  
<xs:element ref="description"/>  
</xs:sequence>  
<xs:attribute name="id" use="required" type="xs:NCName"/>  
</xs:complexType>  
</xs:element>  
<xs:element name="author" type="xs:string"/>  
<xs:element name="title" type="xs:string"/>  
<xs:element name="genre" type="xs:string"/>  
<xs:element name="price" type="xs:decimal"/>  
<xs:element name="publish\_date" type="xs:date"/>  
<xs:element name="description" type="xs:string"/>  
</xs:schema>

# Part #2: create tables

## Step #1: create Oracle directory to store files.

In the last homework I used RTU Oracle database and did not have rights to create an Oracle directory. This time I set up my own Oracle 11g database as a Docker container on the cloud server. Before opening SQL Developer, I first created directory /home/xmlfiles on the server and uploaded my files there.

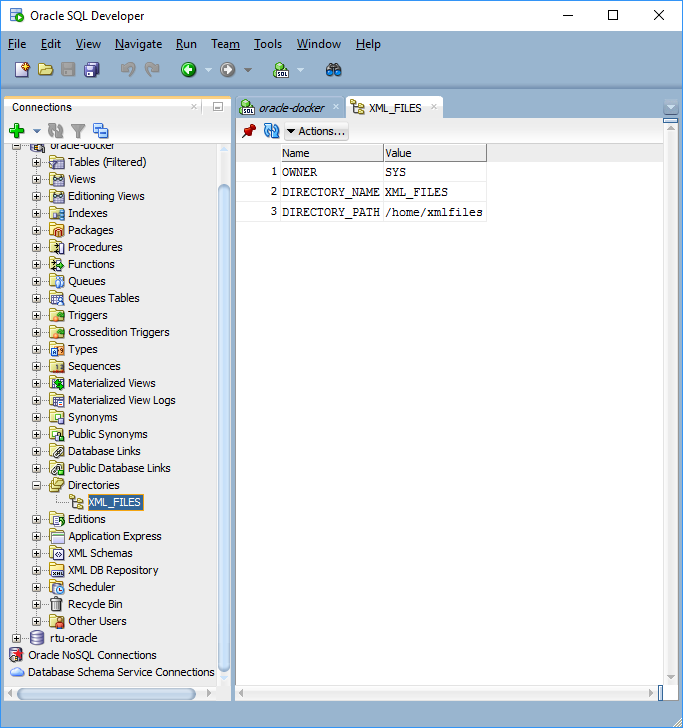


Then I created an Oracle directory pointing to the directory above in SQL Developer.

CREATE OR REPLACE DIRECTORY xml\_files AS '/home/xmlfiles';

Console output:

**Directory XML\_FILES created.**



## Step #2: register schema files.

RegisterSchema procedure is using **BFILE** mechanism to read the source document from a file.

Script for events-schema.xsd:

BEGIN

DBMS\_XMLSCHEMA.registerSchema(

SCHEMAURL => 'EVENTS\_SCHEMA',

SCHEMADOC =>bfilename('XML\_FILES','events-schema.xsd'),

CSID => nls\_charset\_id('AL32UTF8'));

END;

Script output:

**PL/SQL procedure successfully completed.**

Script for catalog-schema.xsd:

BEGIN

DBMS\_XMLSCHEMA.registerSchema(

SCHEMAURL => 'BOOKS\_CATALOG\_SCHEMA',

SCHEMADOC =>bfilename('XML\_FILES','catalog-schema.xsd'),

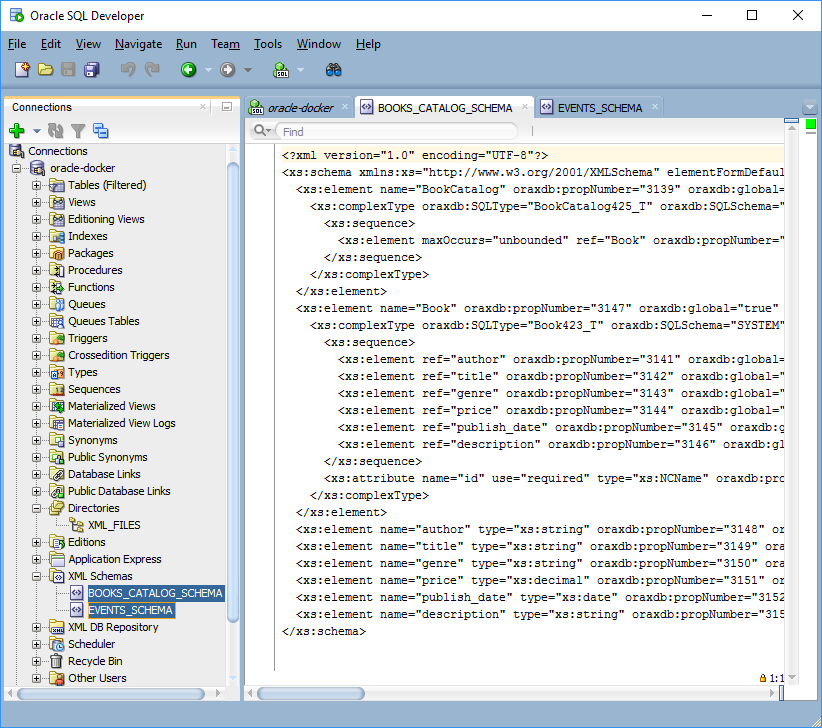
CSID => nls\_charset\_id('AL32UTF8'));

END;

Script output:

**PL/SQL procedure successfully completed.**

Results:



## Step #3: create unstructured table(without schema)

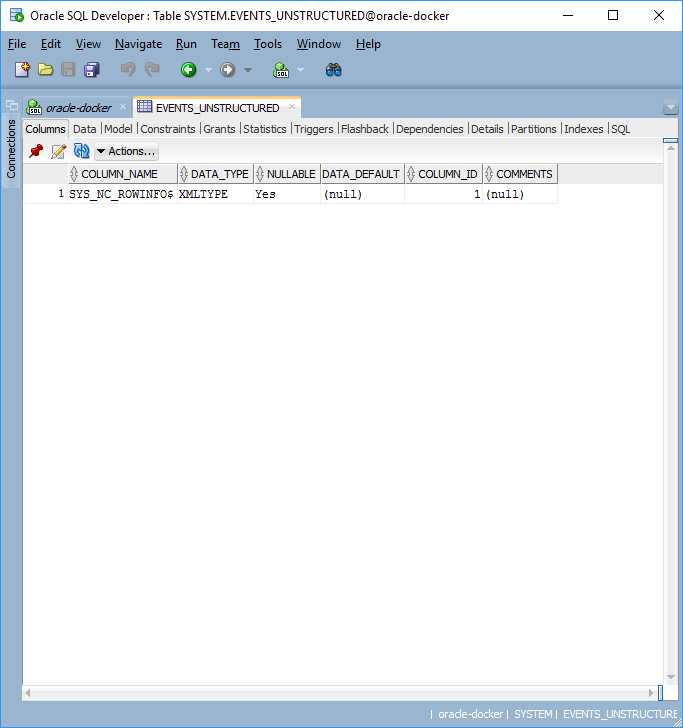
CREATE TABLE events\_unstructured OF XMLTYPE

XMLTYPE STORE AS CLOB;

Script output:

Table EVENTS\_UNSTRUCTURED created.

Result:



## Step #4: create unstructured table(with schema)

CREATE TABLE events\_unstructured\_schema OF XMLTYPE

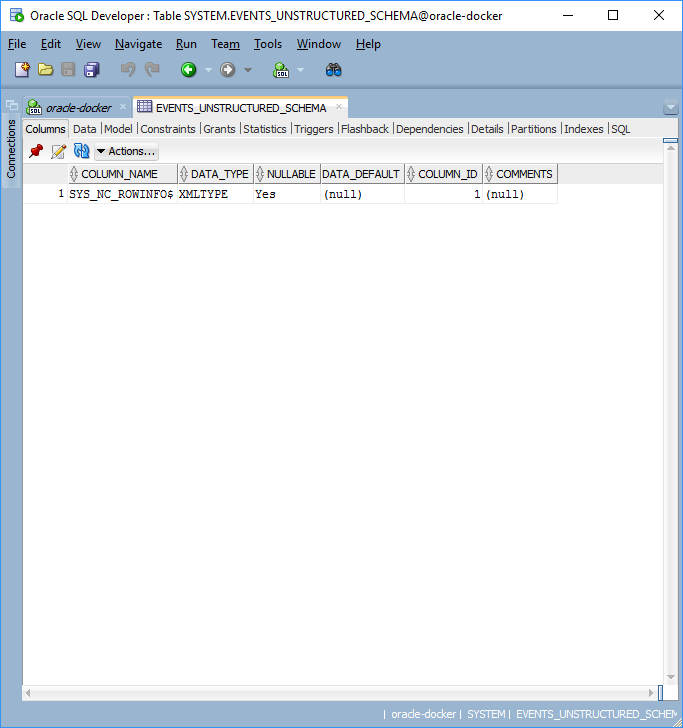
XMLTYPE STORE AS CLOB

XMLSCHEMA "EVENTS\_SCHEMA" ELEMENT "Events";

Script output:

Table EVENTS\_UNSTRUCTURED\_SCHEMA created.

Result:



## Step #5: create structured table

CREATE TABLE books\_structured OF XMLTYPE

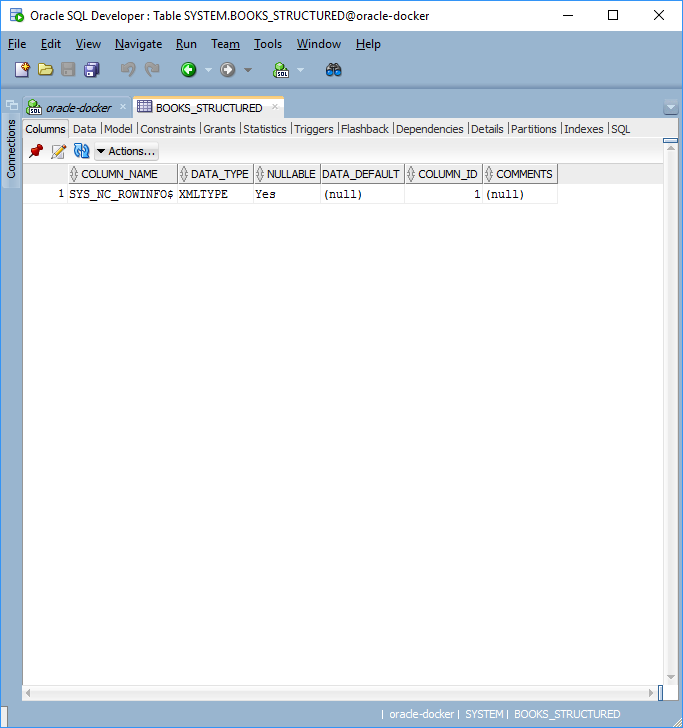
XMLSCHEMA "BOOKS\_CATALOG\_SCHEMA"

ELEMENT "BookCatalog";

Script output:

**Table BOOKS\_STRUCTURED created.**

Result:



# Part #3:input of data

## Step #1: input in unstructured XML storage from file

Before proceeding with this step, I added a file **eventOne.xml**on the server with the following content:

<Eventid="1">  
 <EventDetails>  
 <type>Conference</type>  
 <eventName>Devoxx</eventName>  
 <eventLocation>Antwerpen, Belgium</eventLocation>  
 <participants>1000</participants>  
 <startDate>2016-12-11</startDate>  
 <duration>2</duration>  
 </EventDetails>  
 <Sponsors>  
 <Sponsorid="1">  
 <sponsorName>Oracle</sponsorName>  
 <representative>BrianGoetz</representative>  
 <sponsorRank>Golden</sponsorRank>  
 </Sponsor>  
 <Sponsorid="2">  
 <sponsorName>RedHat</sponsorName>  
 <representative>PeterDavis</representative>  
 <sponsorRank>Silver</sponsorRank>  
 </Sponsor>  
 </Sponsors>  
</Event>

INSERT INTO EVENTS\_UNSTRUCTURED

VALUES(XMLTYPE(bfilename('XML\_FILES','eventOne.xml'), nls\_charset\_id('AL32UTF8')));

Script output:

**1 row inserted.**

## Step #2: input in unstructured XML storage from text

INSERT INTO EVENTS\_UNSTRUCTURED VALUES

(XMLTYPE('<Event id="2">

<EventDetails>

<type>Meetup</type>

<eventName>Berlin JUG</eventName>

<eventLocation>Berlin, Germany</eventLocation>

<participants>55</participants>

<startDate>2016-06-10</startDate>

<duration>1</duration>

</EventDetails>

<Sponsors>

<Sponsor id="3">

<sponsorName>Siemens</sponsorName>

<representative>Julius Shaw</representative>

<sponsorRank>Platinum</sponsorRank>

</Sponsor>

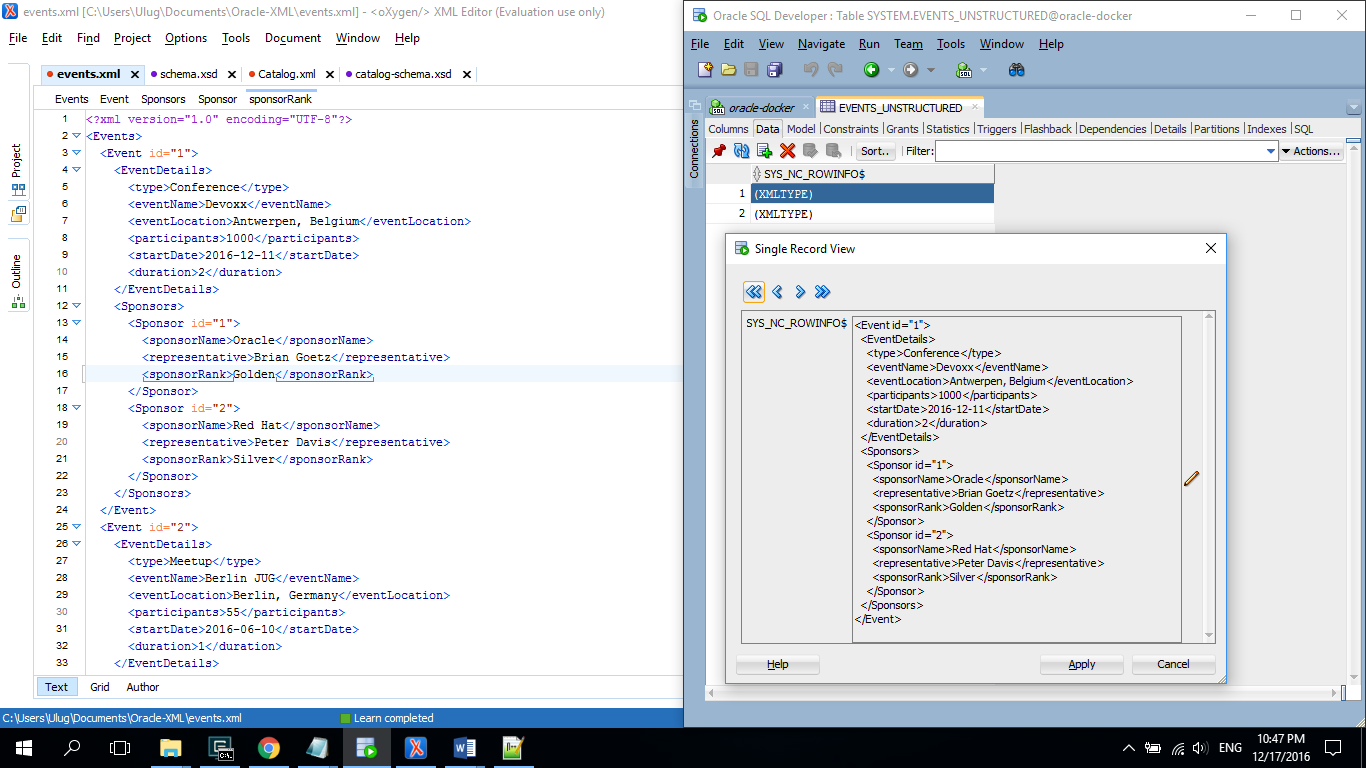
</Sponsors>

</Event>'));

Script output:

**1 row inserted.**

Results:



## Step #3: input in unstructured XML storage with schema from file

Here I will use the file Events.xml shown in the beginning of this work.

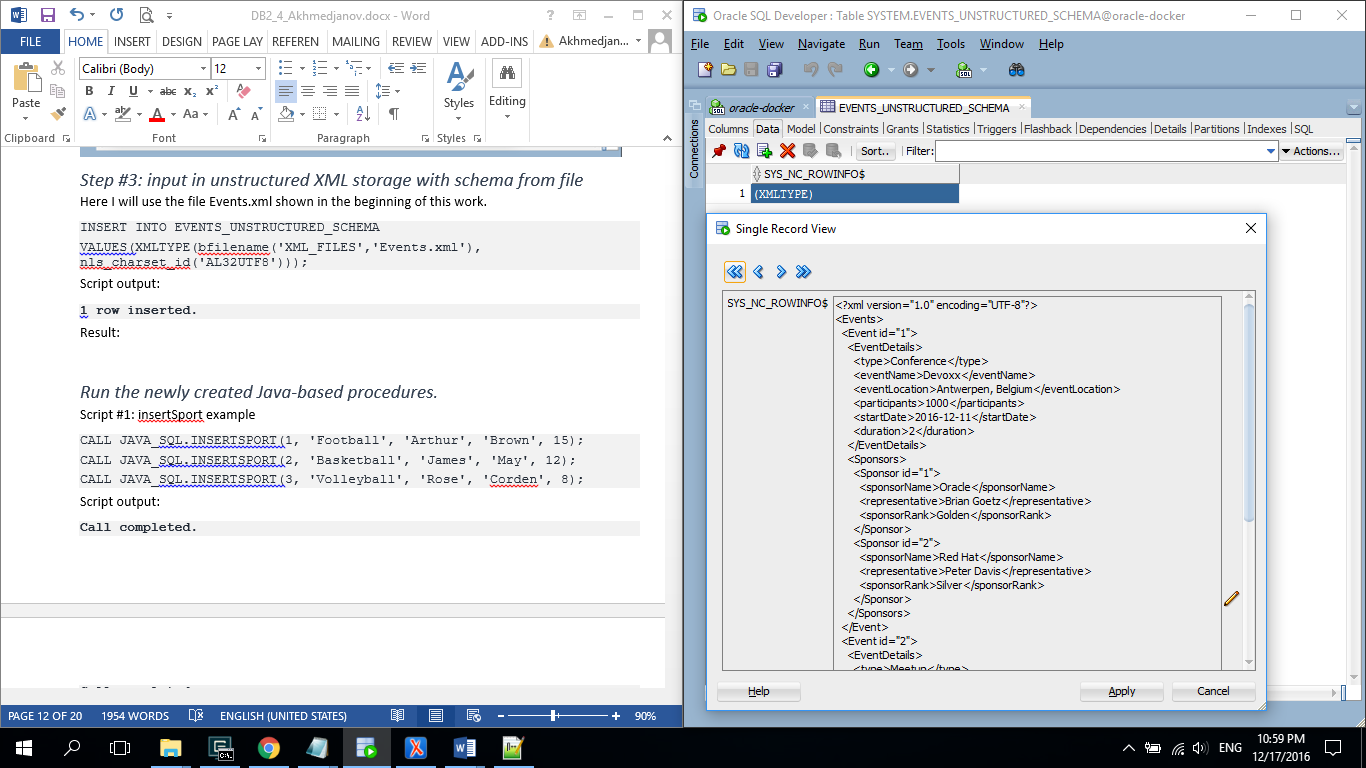
INSERT INTO EVENTS\_UNSTRUCTURED\_SCHEMA

VALUES(XMLTYPE(bfilename('XML\_FILES','Events.xml'), nls\_charset\_id('AL32UTF8')));

Script output:

**1 row inserted.**

Result:



## Step #4: input in structured XML storage with schema from file

In this case, I will use the file Events.xml shown in the beginning of this work.

INSERT INTO BOOKS\_STRUCTURED

VALUES(XMLTYPE(bfilename('XML\_FILES','Catalog.xml'), nls\_charset\_id('AL32UTF8')));

Script output:

1 row inserted.

# Part #4:Data extraction.

## Step #1: EXTRACT() example from unstructured storage without schema + EXISTSNODE()

Here I will select events, where sponsor is Siemens(there is 1 such event).

SELECT EXTRACT(OBJECT\_VALUE, '/Event/EventDetails/type') as Event\_Type,

EXTRACT(OBJECT\_VALUE, '/Event/EventDetails/eventName') as Event\_Name,

EXTRACT(OBJECT\_VALUE, '/Event/EventDetails/participants') as Participants

FROM EVENTS\_UNSTRUCTURED

WHERE EXISTSNODE(OBJECT\_VALUE,

'/Event/Sponsors/Sponsor/sponsorName="Siemens"')=1;

Script output:

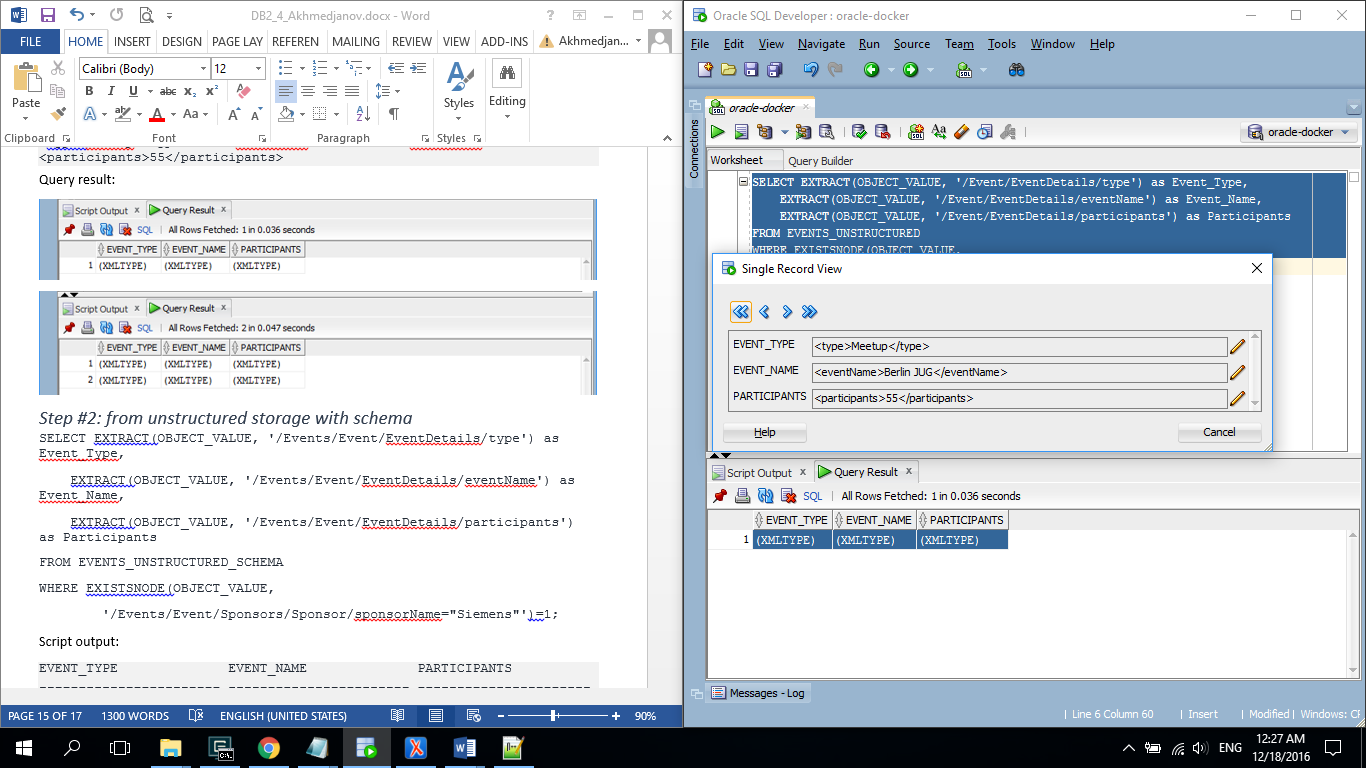
EVENT\_TYPE EVENT\_NAME PARTICIPANTS

----------------------- ----------------------- ----------------------

<type>Conference</type><eventName>Devoxx</eventName><participants>1000</participants>

<type>Meetup</type><eventName>Berlin JUG</eventName><participants>55</participants>

Query result:



## Step #2: EXTRACT() example from unstructured storage with schema

Here I will just show event types, names and number of participants for all events present in the table(there are 2 such events).

SELECT EXTRACT(OBJECT\_VALUE, '/Events/Event/EventDetails/type') as Event\_Type,

EXTRACT(OBJECT\_VALUE, '/Events/Event/EventDetails/eventName') as Event\_Name,

EXTRACT(OBJECT\_VALUE, '/Events/Event/EventDetails/participants') as Participants

FROM EVENTS\_UNSTRUCTURED\_SCHEMA;

Script output:

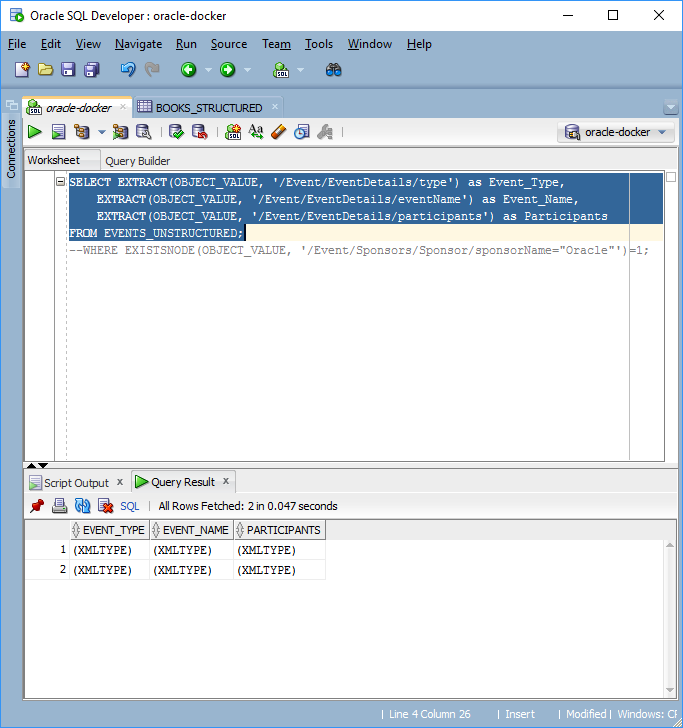
EVENT\_TYPE EVENT\_NAME PARTICIPANTS

----------------------- ----------------------- ----------------------

<type>Conference</type><eventName>Devoxx</eventName><participants>1000</participants>

<type>Meetup</type><eventName>Berlin JUG</eventName><participants>55</participants>

Result:



## Step #3: EXTRACT() example from structured storage

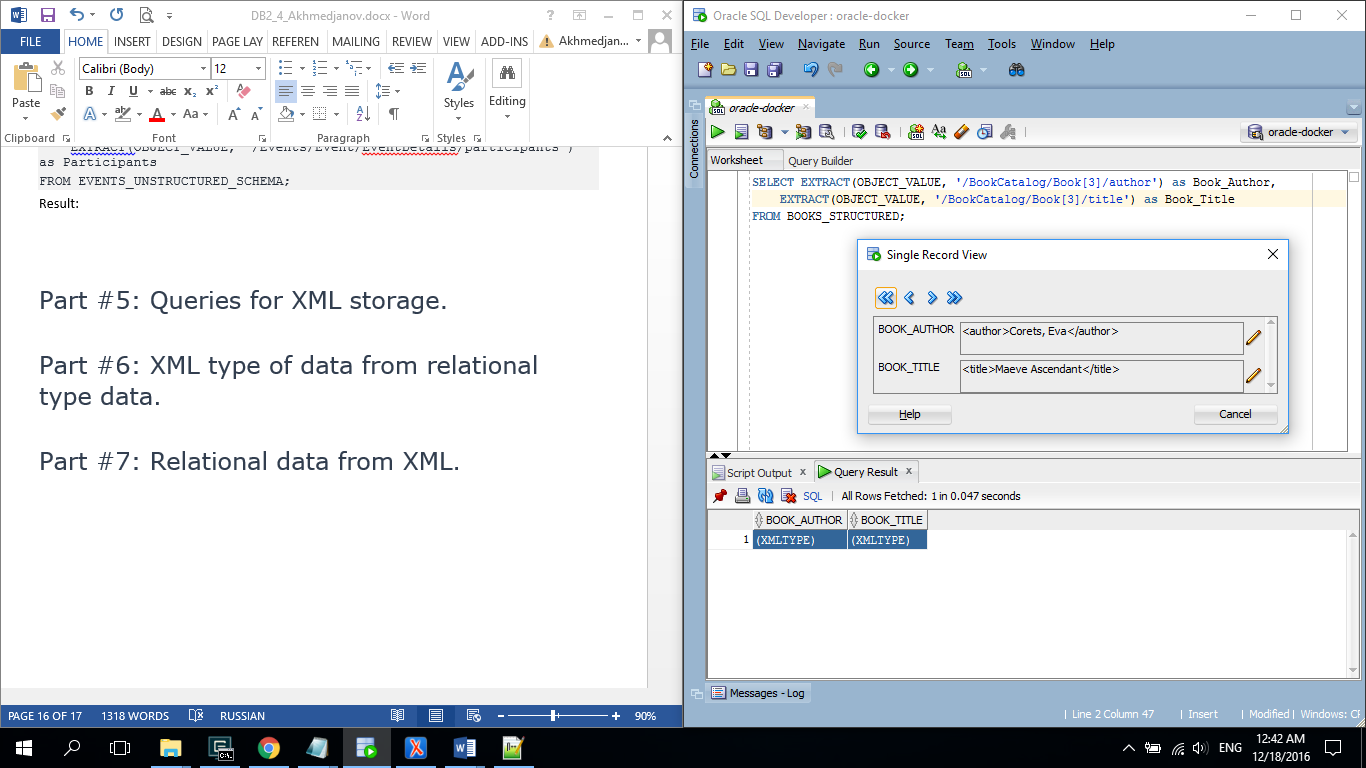
Here I will select the third Book from the BookCatalog file.

SELECT EXTRACT(OBJECT\_VALUE, '/BookCatalog/Book[3]/author/text()') as Book\_Author,

EXTRACT(OBJECT\_VALUE, '/BookCatalog/Book[3]/title/text()') as Book\_Title

FROM BOOKS\_STRUCTURED;

Result:

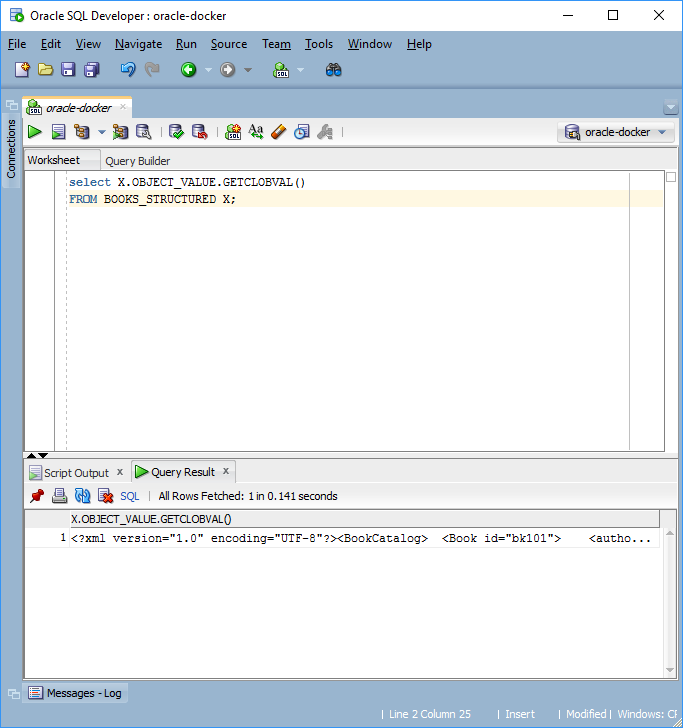


## Step #4: GETCLOBVAL() example

select X.OBJECT\_VALUE.GETCLOBVAL()

FROM BOOKS\_STRUCTURED X;

Result:



## Step #4: EXTRACTVALUE() example for structured storage

Here I will select the first book from BookCatalog and use procedure EXTRACTVALUE().

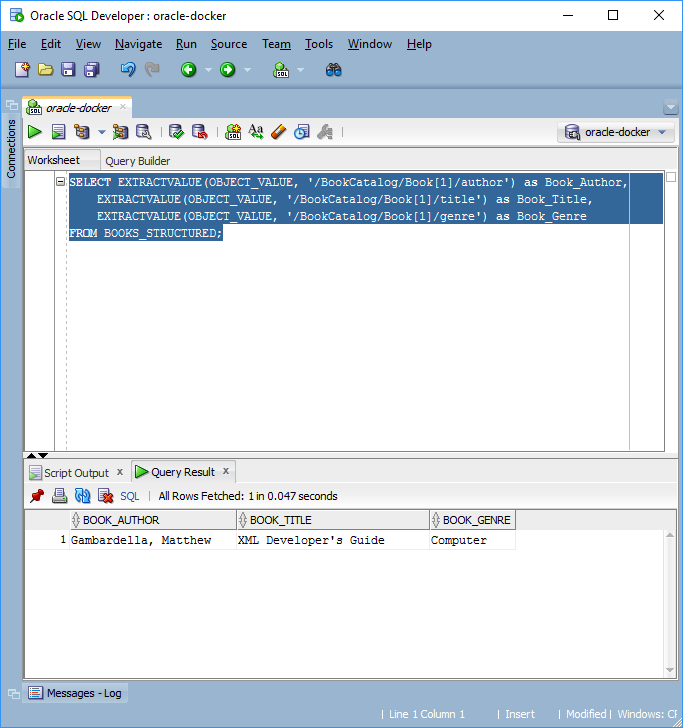
SELECT EXTRACTVALUE(OBJECT\_VALUE, '/BookCatalog/Book[1]/author') as Book\_Author,

EXTRACTVALUE(OBJECT\_VALUE, '/BookCatalog/Book[1]/title') as Book\_Title,

EXTRACTVALUE(OBJECT\_VALUE, '/BookCatalog/Book[1]/genre') as Book\_Genre

FROM BOOKS\_STRUCTURED;

Result:



## Step #5: EXTRACTVALUE() example for unstructured storage

Here I will again select all events, but this time I will use EXTRACTVALUE().

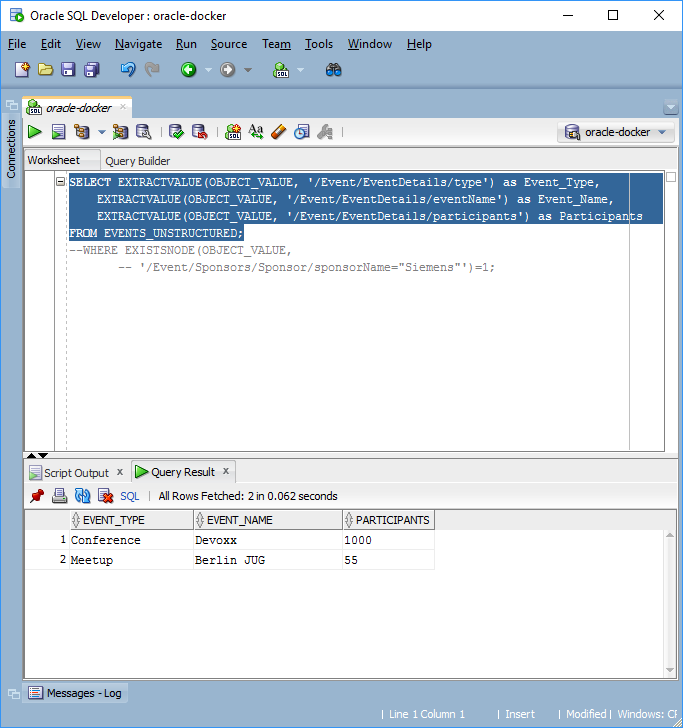
SELECT EXTRACTVALUE(OBJECT\_VALUE, '/Event/EventDetails/type') as Event\_Type,

EXTRACTVALUE(OBJECT\_VALUE, '/Event/EventDetails/eventName') as Event\_Name,

EXTRACTVALUE(OBJECT\_VALUE, '/Event/EventDetails/participants') as Participants

FROM EVENTS\_UNSTRUCTURED;

Result:



# Part #5: XMLQuery examples.

## Step #1: XMLQuery with unstructured storage(without schema)

Here I will look for events where sponsor is Siemens and show its rank.

SELECT XMLQuery(

'for $i in Event/Sponsors

where $i[Sponsor/sponsorName="Siemens"]

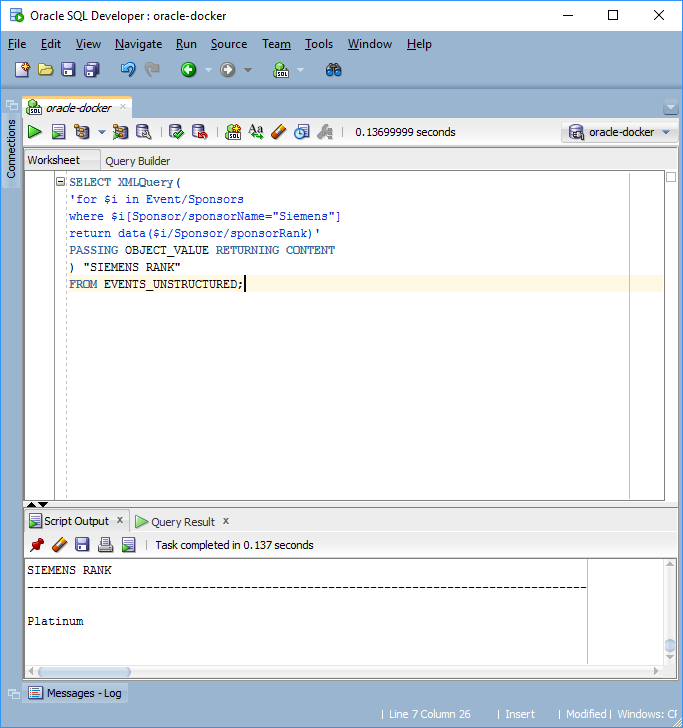
return data($i/Sponsor/sponsorRank)'

PASSING OBJECT\_VALUE RETURNING CONTENT

) "SIEMENS RANK"

FROM EVENTS\_UNSTRUCTURED;

Script output:



## Step #2: XMLQuery with unstructured storage(with schema)

Here I will show events, where number of participants is greater than 100(there is only 1 such event).

SELECT XMLQuery(

'for $i in Events/Event

where $i[EventDetails/participants>100]

return data($i/EventDetails/eventName)'

PASSING OBJECT\_VALUE RETURNING CONTENT

) "BIG EVENTS"

FROM EVENTS\_UNSTRUCTURED\_SCHEMA;

Script output:

BIG EVENTS

----------------------------------------------------------------------

Devoxx

## Step #3: XMLQuery with structured storage

Here I will look for all books, where genre is ‘Fantasy’(there are 4 such books).

SELECT XMLQuery(

'for $i in BookCatalog/Book

where $i[genre="Fantasy"]

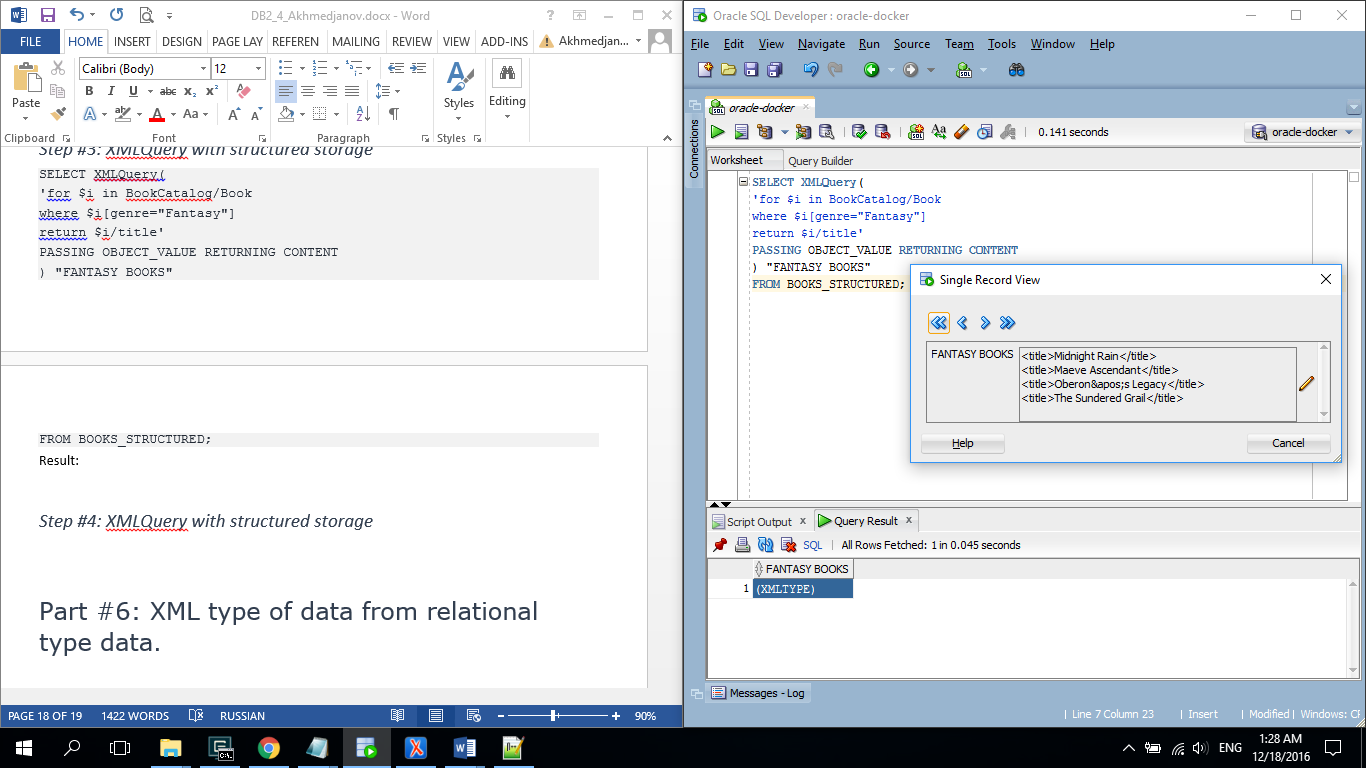
return $i/title'

PASSING OBJECT\_VALUE RETURNING CONTENT

) "FANTASY BOOKS"

FROM BOOKS\_STRUCTURED;

Result:



## Step #4: XMLQuery with structured storage

Here I will show all books with price greater than 20(there is only one such book in XML file).

SELECT XMLQuery(

'for $i in BookCatalog/Book

where $i[price>20]

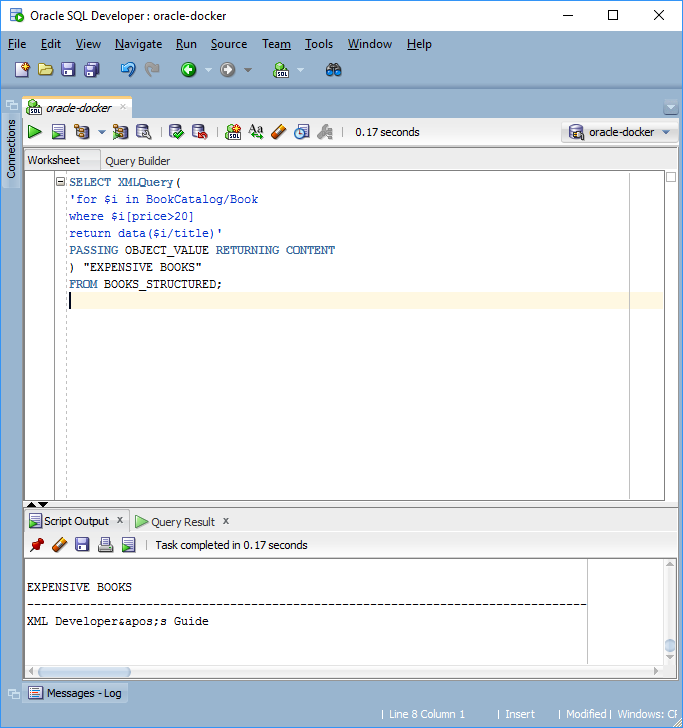
return data($i/title)'

PASSING OBJECT\_VALUE RETURNING CONTENT

) "EXPENSIVE BOOKS"

FROM BOOKS\_STRUCTURED;

Result:



# Part #6: XML type of data from relational type data.

First, I created the table Books

CREATE TABLE BOOKS(

book\_id NUMBER PRIMARY KEY,

author VARCHAR2(50),

title VARCHAR2(50),

genre VARCHAR2(50),

price NUMBER,

publish\_date DATE);

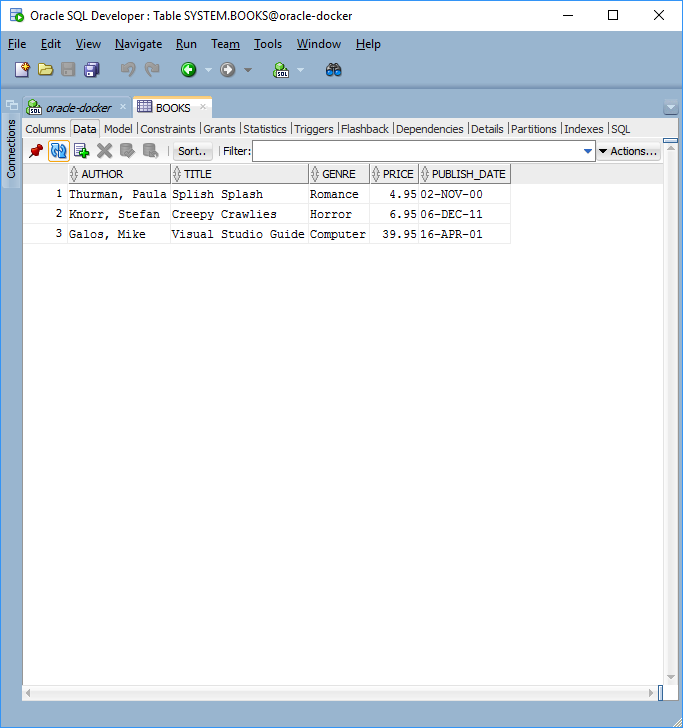
Then, inserted a few rows

INSERT INTO BOOKS VALUES(1, 'Thurman, Paula','Splish Splash','Romance',4.95,'02-Nov-2000');

INSERT INTO BOOKS VALUES(2, 'Knorr, Stefan','Creepy Crawlies','Horror',6.95,'06-Dec-2011');

INSERT INTO BOOKS VALUES(3, 'Galos, Mike','Visual Studio Guide','Computer',39.95,'16-Apr-2001');

Result:



Now I will generate XML data from this table using the following query:

SELECT XMLElement("Book", XMLAttributes(B.BOOK\_ID as "id"),

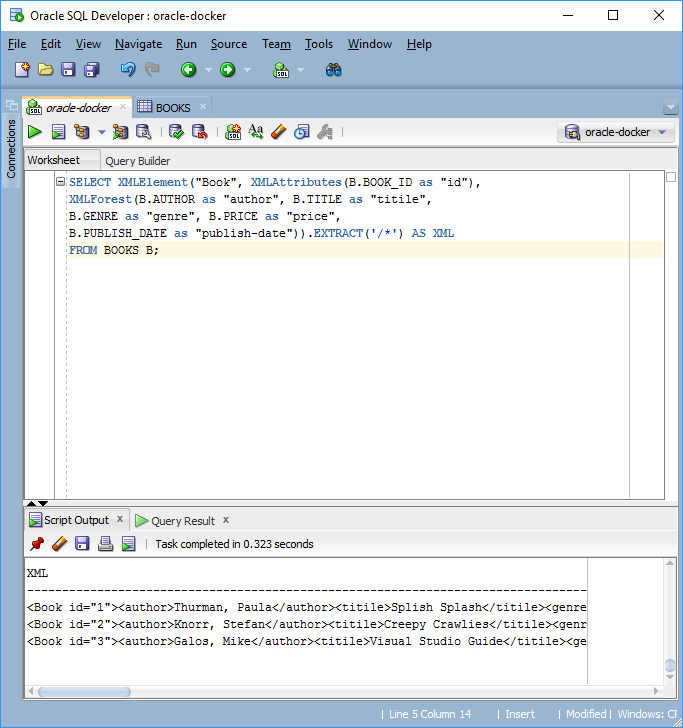
XMLForest(B.AUTHOR as "author", B.TITLE as "titile",

B.GENRE as "genre", B.PRICE as "price",

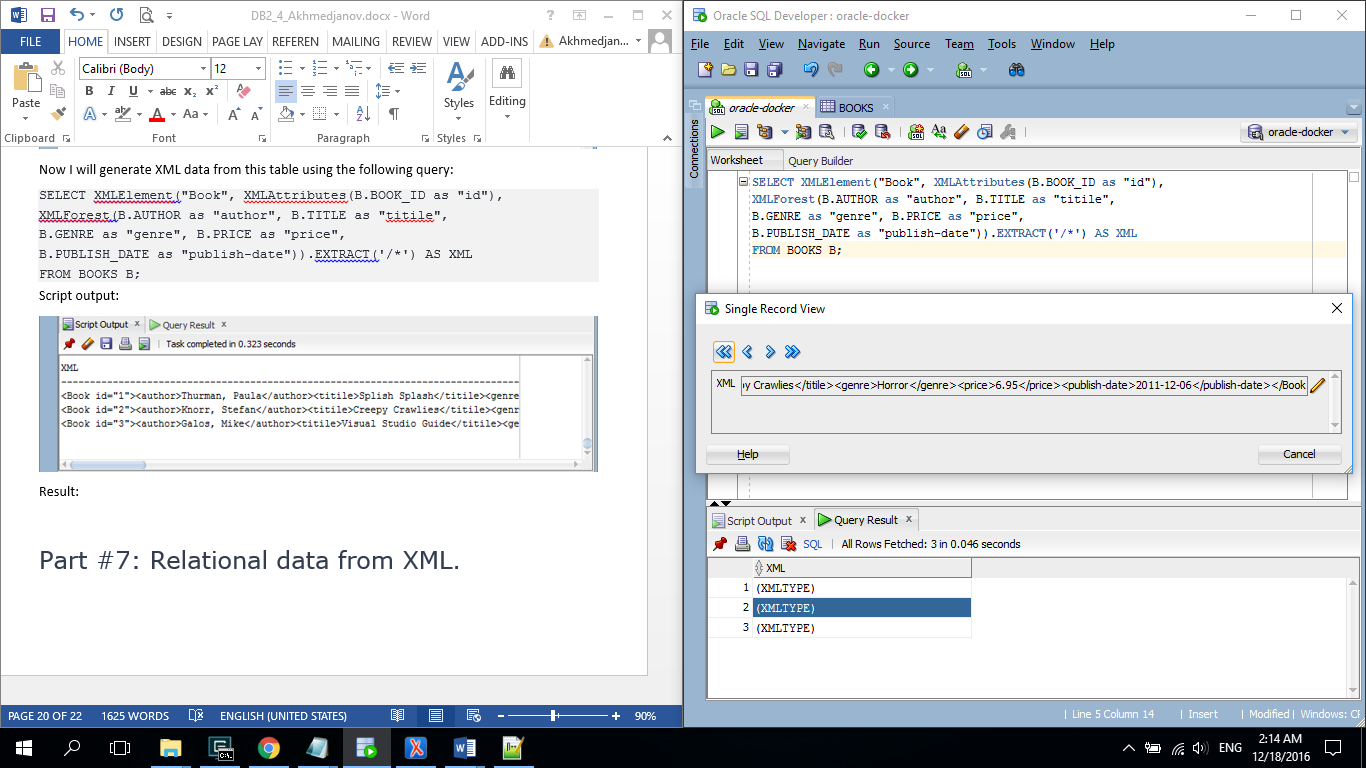
B.PUBLISH\_DATE as "publish-date")).EXTRACT('/\*') AS XML

FROM BOOKS B;

Script output:



As we expected, exactly 3 XMLs were generated. To see the whole XML document, we could use **Single Record View** in **Query Result** tab:



# Part #7: Relational data from XML.

In this part, I will again create a table for Books.

CREATE TABLE BOOKS\_FROM\_XML(

author VARCHAR2(50),

title VARCHAR2(50),

genre VARCHAR2(50),

price NUMBER,

publish\_dateVARCHAR2(15),

description VARCHAR2(255));

Now I will write a procedure to translate XML data type to relational type.

CREATE OR REPLACE PROCEDURE LOADBOOKS(FILENAME IN VARCHAR2) AS

books\_filebfile;

xml\_dataclob;

xml\_handleDBMS\_XMLSTORE.ctxType;

row\_num number;

varxmlType;

csid integer;

dst\_offset number :=1;

src\_offset number :=1;

lang\_ctx number :=dbms\_lob.default\_lang\_ctx;

warning number;

begin

books\_file := bfilename('XML\_FILES', FILENAME);

selectnls\_charset\_id('al32UTF8') into csid from dual;

DBMS\_LOB.CREATETEMPORARY(xml\_data, TRUE);

DBMS\_LOB.FILEOPEN(books\_file, DBMS\_LOB.FILE\_READONLY);

DBMS\_LOB.LOADCLOBFROMFILE(xml\_data, books\_file, DBMS\_LOB.GETLENGTH(books\_file),

DST\_OFFSET, SRC\_OFFSET, CSID, LANG\_CTX, WARNING);

DBMS\_LOB.FILECLOSE(books\_file);

xml\_handle := DBMS\_XMLSTORE.newContext('BOOKS\_FROM\_XML');

DBMS\_XMLGEN.setConvertSpecialChars(xml\_handle, true);

DBMS\_XMLSTORE.clearUpdateColumnList(xml\_handle);

DBMS\_XMLSTORE.setRowTag(xml\_handle,'Book');

DBMS\_XMLSTORE.setUpdateColumn(xml\_handle, 'author');

DBMS\_XMLSTORE.setUpdateColumn(xml\_handle, 'title');

DBMS\_XMLSTORE.setUpdateColumn(xml\_handle, 'genre');

DBMS\_XMLSTORE.setUpdateColumn(xml\_handle, 'price');

DBMS\_XMLSTORE.setUpdateColumn(xml\_handle, 'publish\_date');

DBMS\_XMLSTORE.setUpdateColumn(xml\_handle, 'description');

row\_num:= DBMS\_XMLSTORE.insertXML(xml\_handle, xml\_data);

DBMS\_OUTPUT.PUT\_LINE(row\_num || ' rows inserted.' );

DBMS\_XMLSTORE.closeContext(xml\_handle);

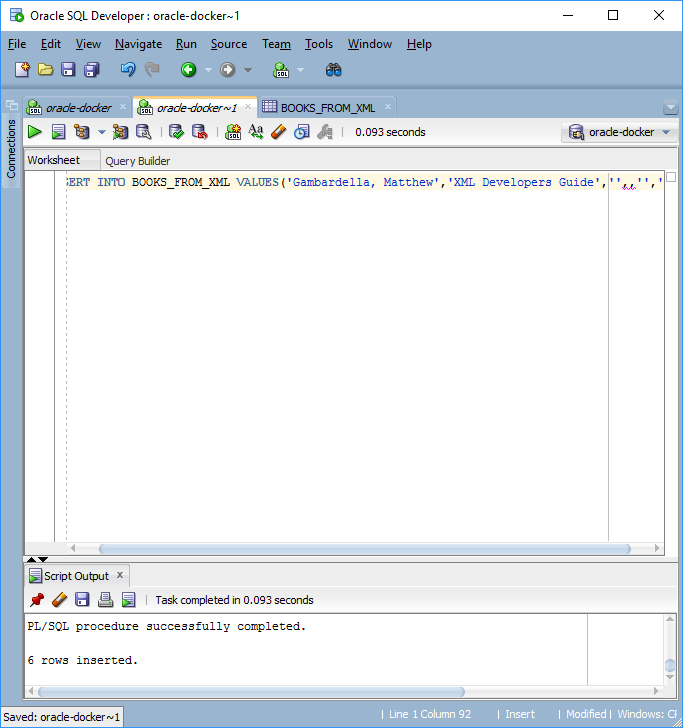
DBMS\_LOB.freeTemporary(xml\_data);

end LOADBOOKS;

Then I call the newly created procedure with filename in the argument.

execute LOADBOOKS('Catalog.xml');

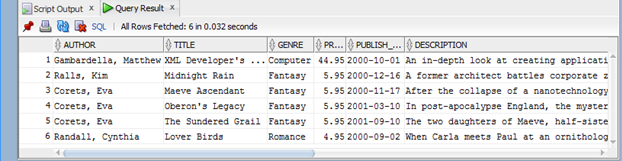
Script output:



Result:

Now I will verify that data was indeed inserted in the table BOOKS\_FROM\_XML.

SELECT \* FROM BOOKS\_FROM\_XML;



# Conclusions:

To me, this work was more difficult than all previous works. Firstly, I was unable to create a directory in the RTU database, therefore I set up my own database on the cloud server. Secondly, when working with XML files in Oracle database, it is sometimes very difficult to understand why something went wrong. Because error messages are not very informative and there is no way to debug the procedure.

In general, this work improved my skills of working with XMLs and schema files. I wish I had Oracle 12 database to learn how JSON storage works in Oracle, because this format is very popular in modern Web.