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
Part #1: create XMLs and schema files(XSD) 4
Part #2: create tables 9
Part #3: input of data 11
Part #4: Data extraction. 14
Part #5: XMLQuery examples. 17
Part #6: XML type of data from relational type data. 19
Part #7: Relational data from XML. 20
Conclusions: 22

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>
```

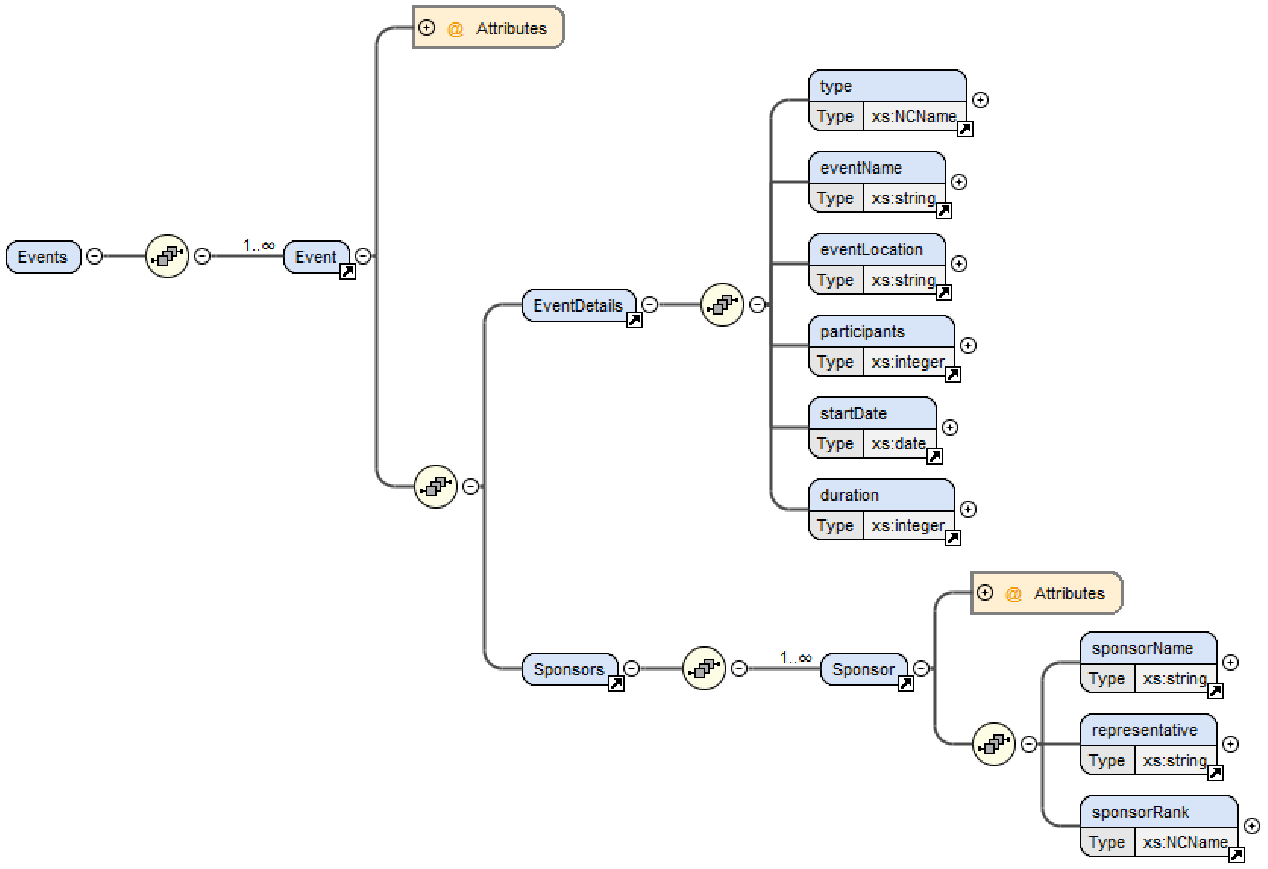
XML document #2, Catalog.xml

```
<?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.

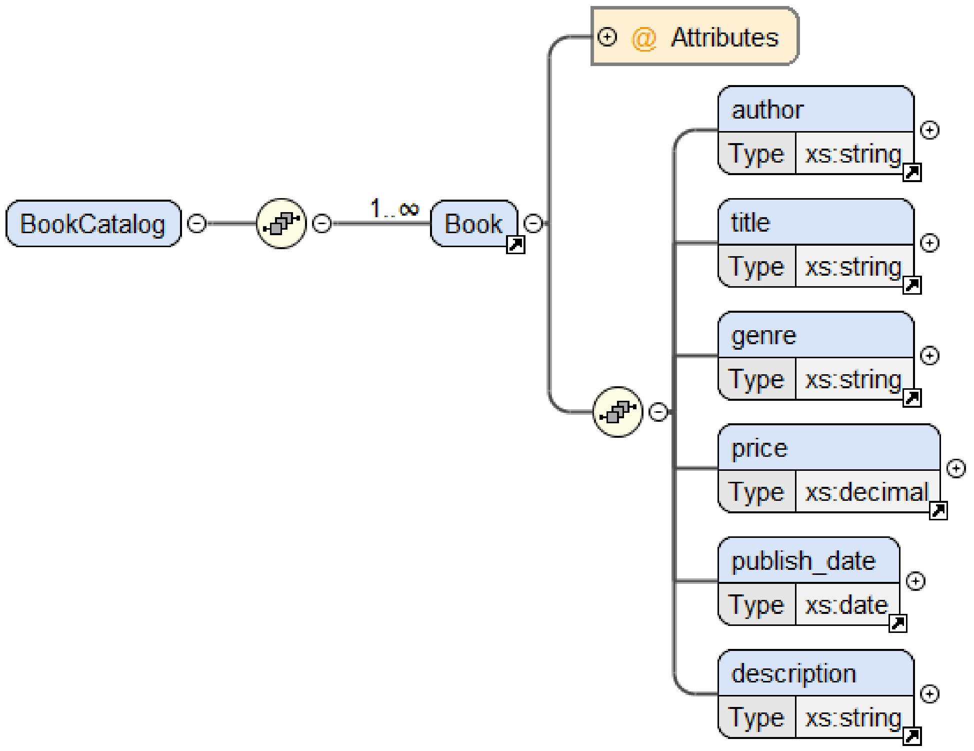


```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schemaxmlns:xs="http://www.w3.org/2001/XMLSchema"elementFormDefault="qualified">
<xs:element name="Events">
<xs:complexType>
<xs:sequence>
<xs:elementmaxOccurs="unbounded" ref="Event"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="Event">
<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.

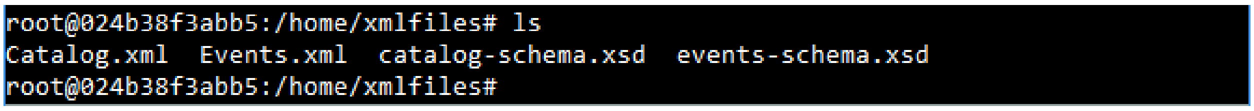

Attributes
author
Type xs:string
title
Type xs:string
genre
Type xs:string
price
Type xs:decimal
publish_date
Type xs:date
description
Type xs:string
BookCatalog
1..∞
Book

```
<?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.



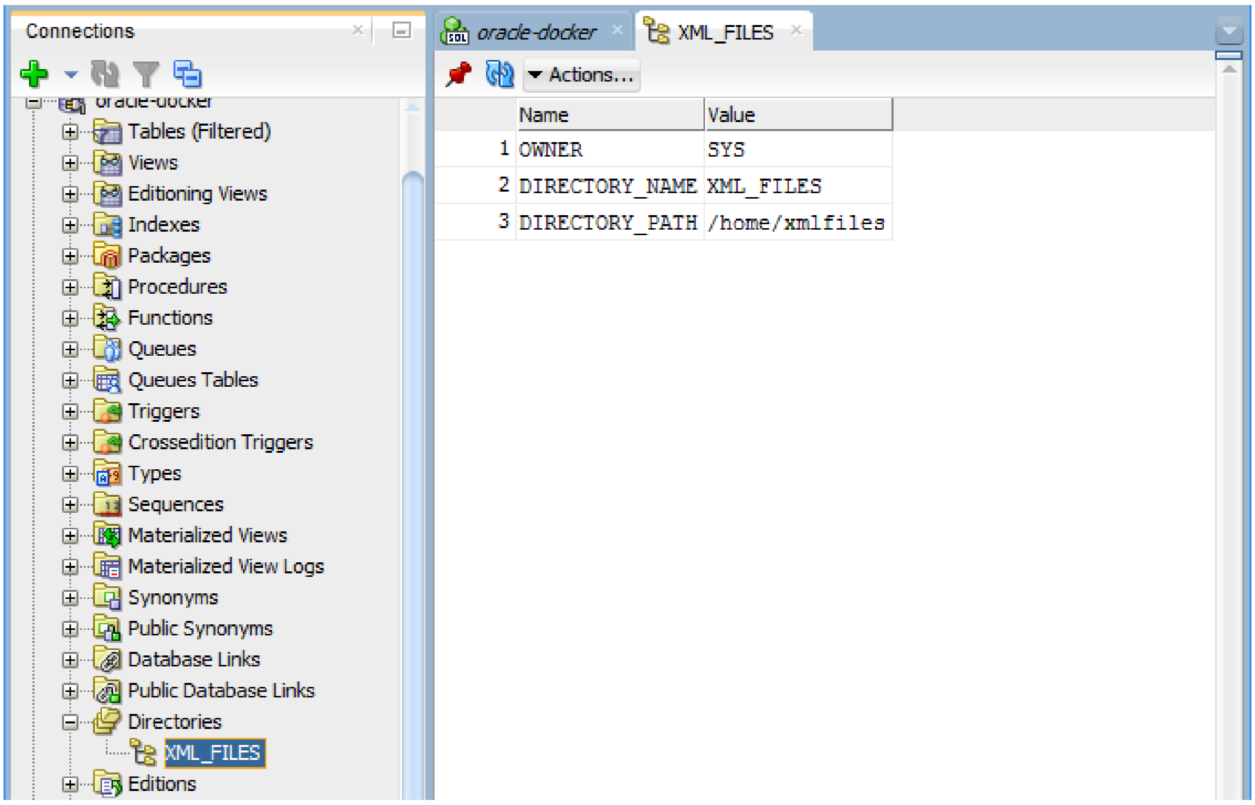
```
root@024b38f3abb5:/home/xmlfiles# ls
Catalog.xml  Events.xml  catalog-schema.xsd  events-schema.xsd
root@024b38f3abb5:/home/xmlfiles#
```

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.
```



	Name	Value
1	OWNER	SYS
2	DIRECTORY_NAME	XML_FILES
3	DIRECTORY_PATH	/home/xmlfiles

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.

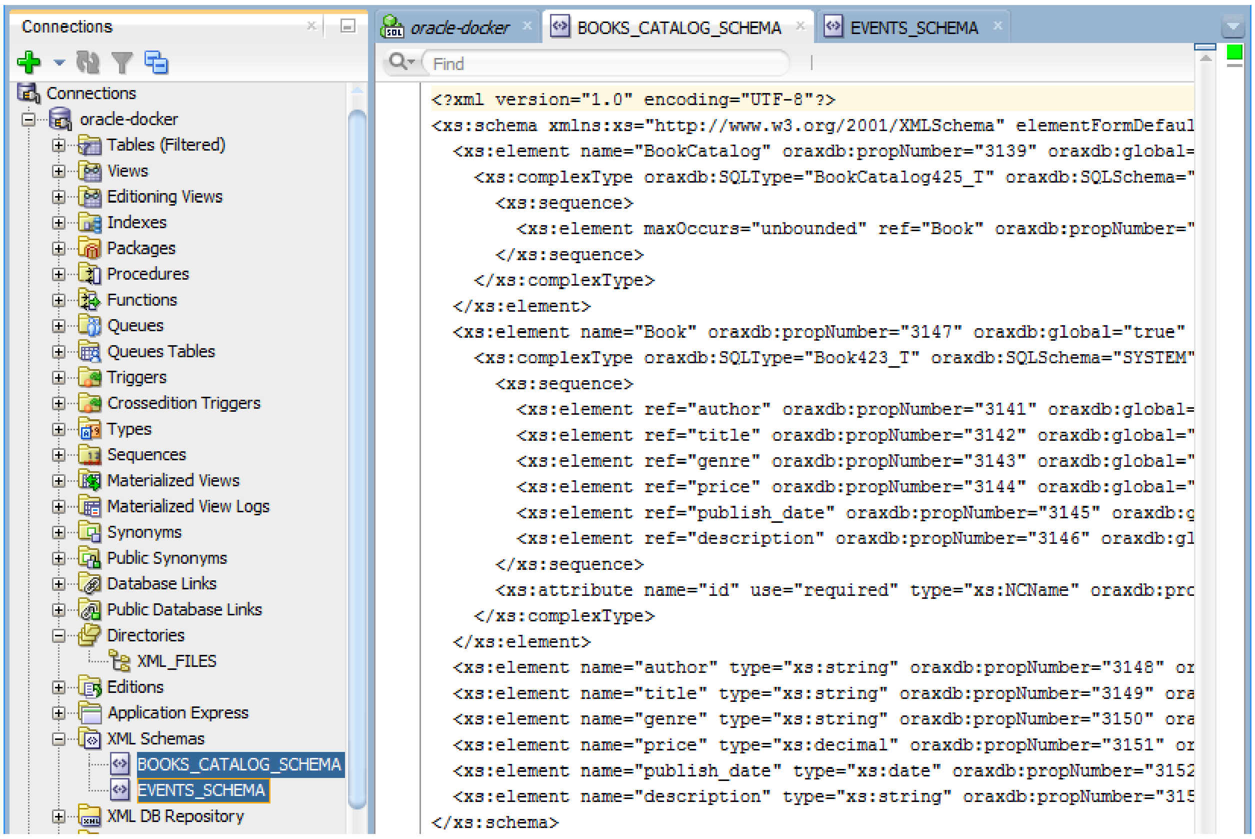
Script for catalog-schema.xsd:

```
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:



```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefaul
  <xs:element name="BookCatalog" oraxdb:propNumber="3139" oraxdb:global=
    <xs:complexType oraxdb:SQLType="BookCatalog425_T" oraxdb:SQLSchema="
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="Book" oraxdb:propNumber="
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Book" oraxdb:propNumber="3147" oraxdb:global="true"
    <xs:complexType oraxdb:SQLType="Book423_T" oraxdb:SQLSchema="SYSTEM"
      <xs:sequence>
        <xs:element ref="author" oraxdb:propNumber="3141" oraxdb:global=
        <xs:element ref="title" oraxdb:propNumber="3142" oraxdb:global="
        <xs:element ref="genre" oraxdb:propNumber="3143" oraxdb:global="
        <xs:element ref="price" oraxdb:propNumber="3144" oraxdb:global="
        <xs:element ref="publish_date" oraxdb:propNumber="3145" oraxdb:g
        <xs:element ref="description" oraxdb:propNumber="3146" oraxdb:gl
      </xs:sequence>
      <xs:attribute name="id" use="required" type="xs:NCName" oraxdb:pro
    </xs:complexType>
  </xs:element>
  <xs:element name="author" type="xs:string" oraxdb:propNumber="3148" or
  <xs:element name="title" type="xs:string" oraxdb:propNumber="3149" ora
  <xs:element name="genre" type="xs:string" oraxdb:propNumber="3150" ora
  <xs:element name="price" type="xs:decimal" oraxdb:propNumber="3151" or
  <xs:element name="publish_date" type="xs:date" oraxdb:propNumber="3152
  <xs:element name="description" type="xs:string" oraxdb:propNumber="315
</xs:schema>
```

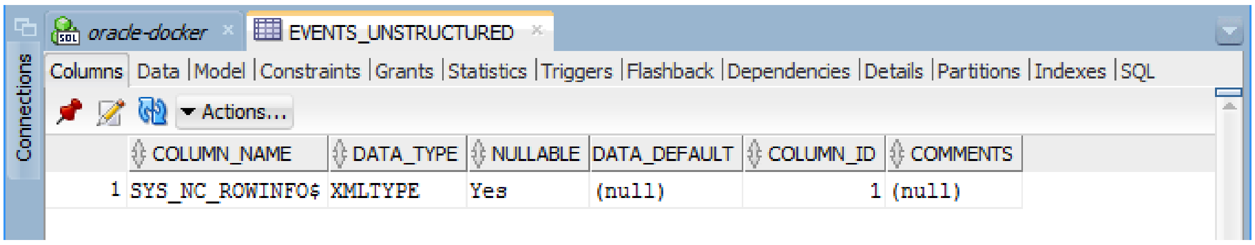
Step #3: create unstructured table(without schema)

```
CREATE TABLE events_unstructured OF XMLTYPE
XMLTYPE STORE AS CLOB;
```

Script output:

```
Table EVENTS_UNSTRUCTURED created.
```

Result:


oracle-docker
EVENTS_UNSTRUCTURED
Connections
Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL
Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	SYS_NC_ROWINFO$	XMLTYPE	Yes	(null)	1	(null)

Step #4: create unstructured table(with schema)

CREATE TABLE events_unstructured_schema OF XMLTYPE

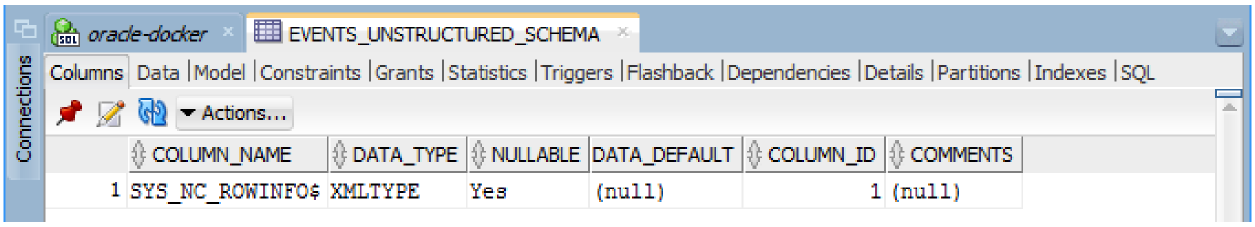
XMLTYPE STORE AS CLOB

XMLSCHEMA "EVENTS_SCHEMA" ELEMENT "Events";

Script output:

```
Table EVENTS_UNSTRUCTURED_SCHEMA created.
```

Result:


oracle-docker
EVENTS_UNSTRUCTURED_SCHEMA
Connections
Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL
Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	SYS_NC_ROWINFO$	XMLTYPE	Yes	(null)	1	(null)

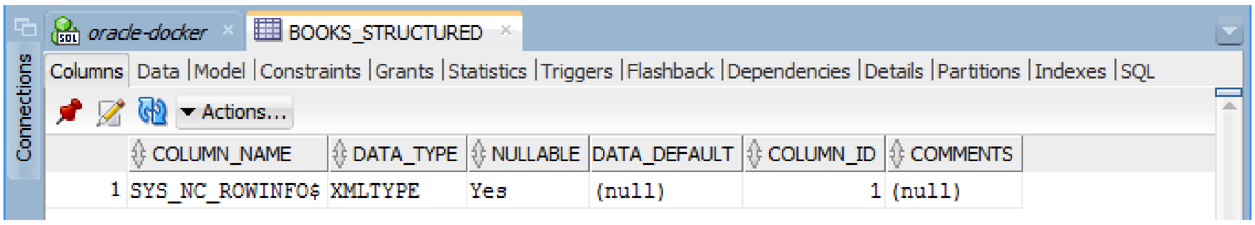
Step #5: create structured table

```
CREATE TABLE books_structured OF XMLTYPE
XMLSCHEMA "BOOKS_CATALOG_SCHEMA"
ELEMENT "BookCatalog";
```

Script output:

```
Table BOOKS_STRUCTURED created.
```

Result:


oracle-docker
BOOKS_STRUCTURED
Connections
Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL
Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	SYS_NC_ROWINFO$	XMLTYPE	Yes	(null)	1	(null)

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.
```

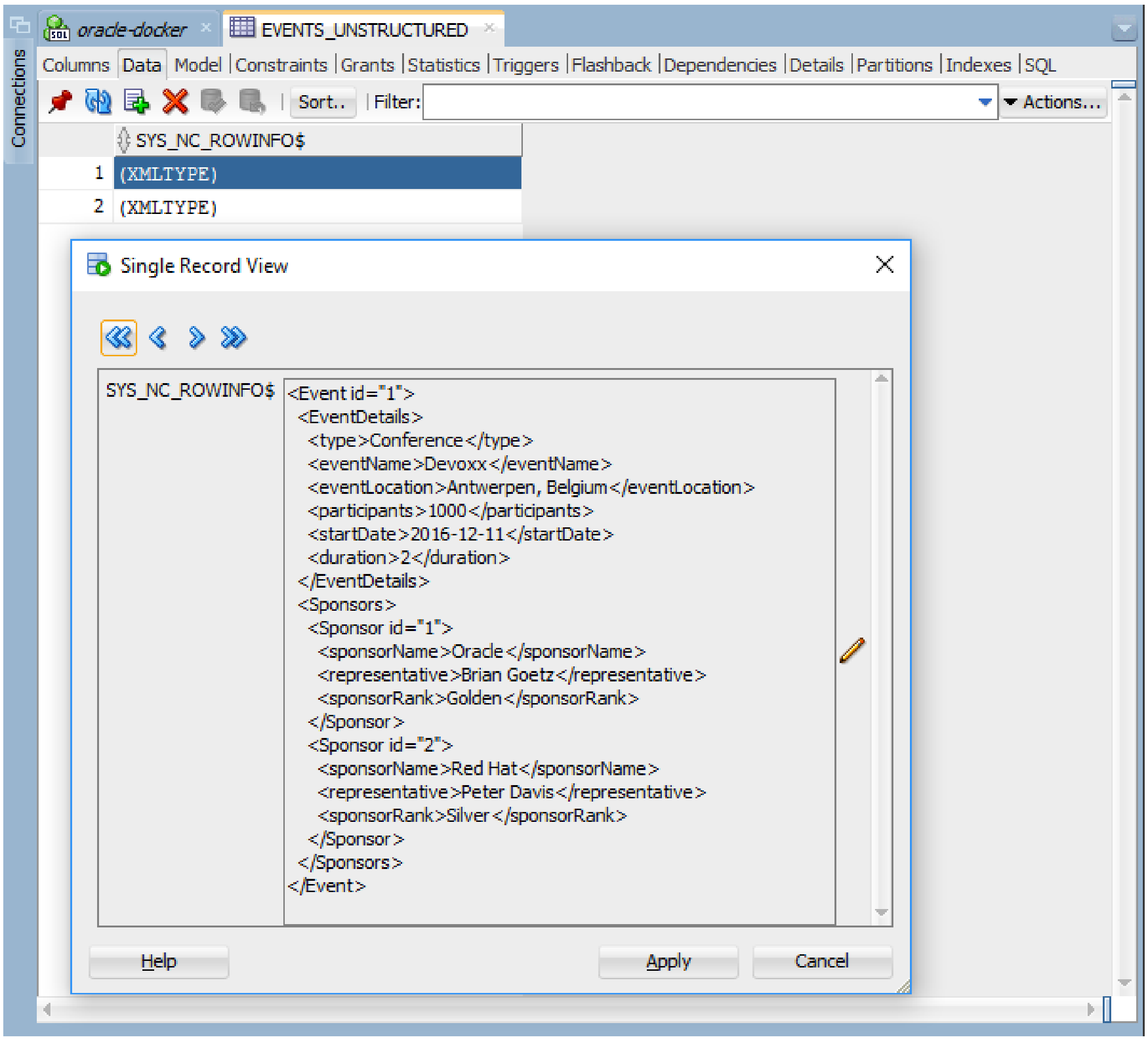
Step #2: input in unstructured XML storage from text

```
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:



```
<Event id="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>
    <Sponsor id="1">
      <sponsorName>Oracle</sponsorName>
      <representative>Brian Goetz</representative>
      <sponsorRank>Golden</sponsorRank>
    </Sponsor>
    <Sponsor id="2">
      <sponsorName>Red Hat</sponsorName>
      <representative>Peter Davis</representative>
      <sponsorRank>Silver</sponsorRank>
    </Sponsor>
  </Sponsors>
</Event>
```

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

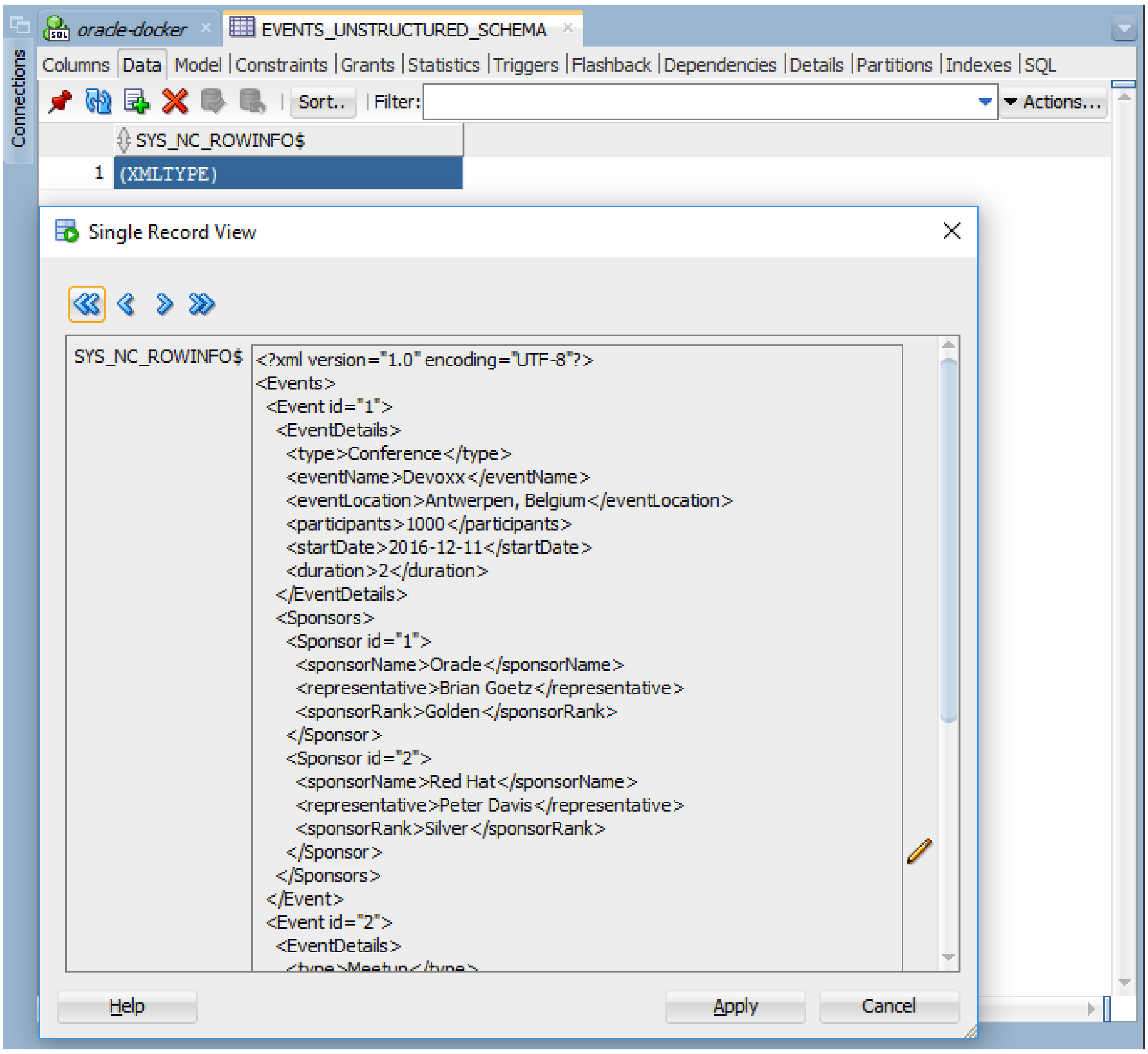
Here I will use the file Events.xml shown in the beginning of this work.

```
INSERT INTO EVENTS_UNSTRUCTURED_SCHEMA
VALUES(XMLTYPE(bfilename('XML_FILES','Events.xml'), nls_charset_id('AL32UTF8')));
```

Script output:

```
1 row inserted.
```

Result:


oracle-docker
EVENTS_UNSTRUCTURED_SCHEMA
Connections
Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL
Sort..
Filter:
Actions...
SYS_NC_ROWINFO$
1 (XMLTYPE)
Single Record View
SYS_NC_ROWINFO$
Help
Apply
Cancel

```
<?xml version="1.0" encoding="UTF-8"?>
<Events>
 <Event id="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>
   <Sponsor id="1">
    <sponsorName>Oracle</sponsorName>
    <representative>Brian Goetz</representative>
    <sponsorRank>Golden</sponsorRank>
   </Sponsor>
   <Sponsor id="2">
    <sponsorName>Red Hat</sponsorName>
    <representative>Peter Davis</representative>
    <sponsorRank>Silver</sponsorRank>
   </Sponsor>
  </Sponsors>
 </Event>
 <Event id="2">
  <EventDetails>
   <type>Meetup</type>
```

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()

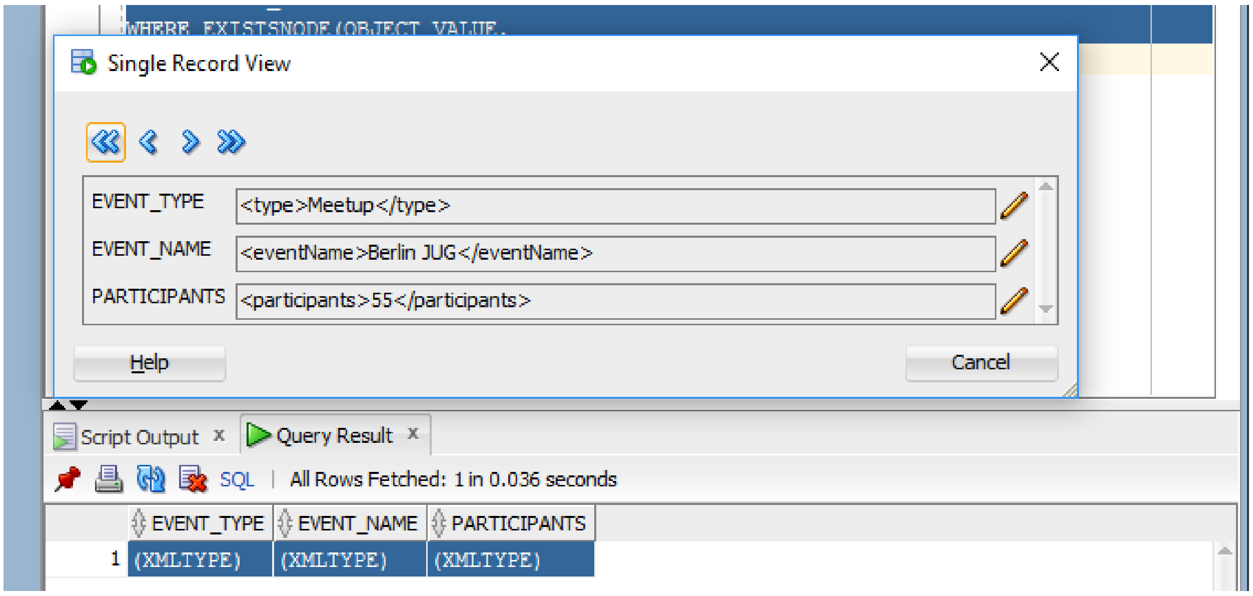
Here I will select events, where sponsor is Siemens(there is 1 such event).

```
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:



```
WHERE EXISTSNODE(OBJECT_VALUE,
```

EVENT_TYPE	<type>Meetup</type>
EVENT_NAME	<eventName>Berlin JUG</eventName>
PARTICIPANTS	<participants>55</participants>

All Rows Fetched: 1 in 0.036 seconds

	EVENT_TYPE	EVENT_NAME	PARTICIPANTS
1	(XMLTYPE)	(XMLTYPE)	(XMLTYPE)

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

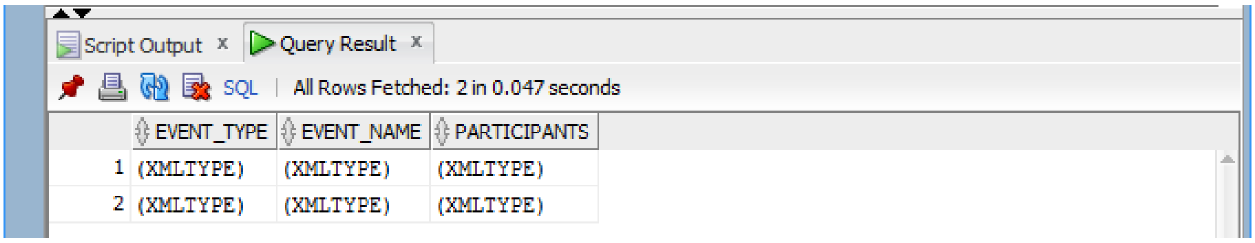
Here I will just show event types, names and number of participants for all events present in the table(there are 2 such events).

```
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:



All Rows Fetched: 2 in 0.047 seconds

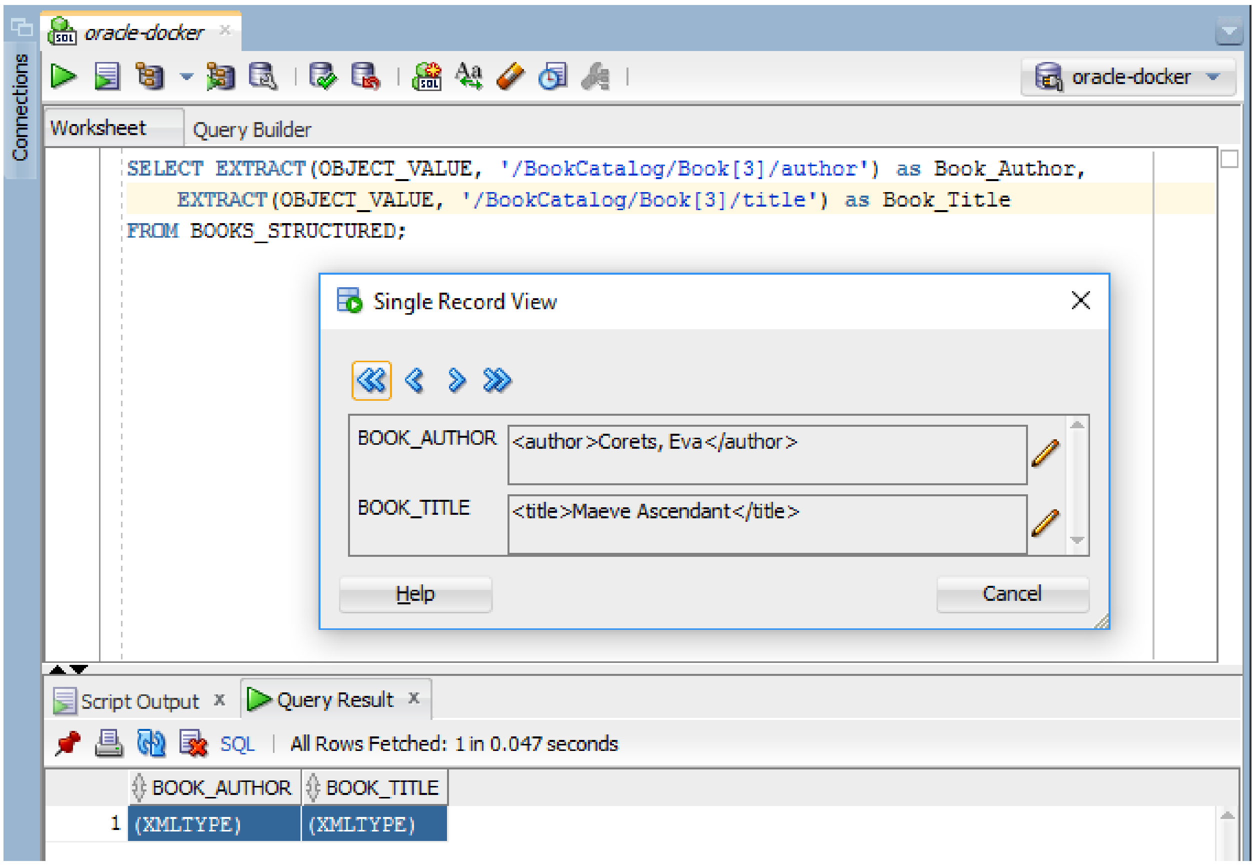
	EVENT_TYPE	EVENT_NAME	PARTICIPANTS
1	(XMLTYPE)	(XMLTYPE)	(XMLTYPE)
2	(XMLTYPE)	(XMLTYPE)	(XMLTYPE)

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:



```
SELECT EXTRACT(OBJECT_VALUE, '/BookCatalog/Book[3]/author') as Book_Author,
    EXTRACT(OBJECT_VALUE, '/BookCatalog/Book[3]/title') as Book_Title
FROM BOOKS_STRUCTURED;
```

Single Record View

BOOK_AUTHOR	<author>Corets, Eva</author>
BOOK_TITLE	<title>Maeve Ascendant</title>

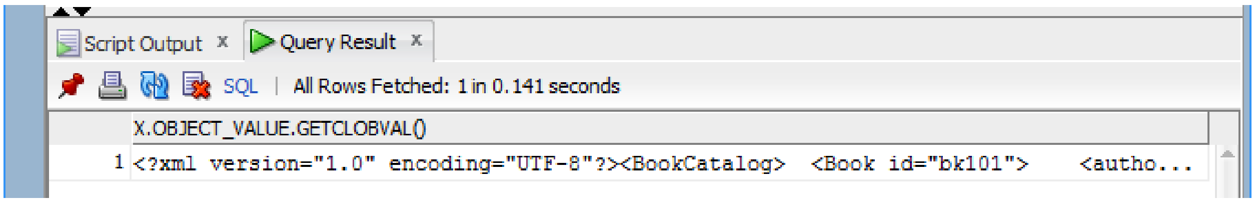
All Rows Fetched: 1 in 0.047 seconds

	BOOK_AUTHOR	BOOK_TITLE
1	(XMLTYPE)	(XMLTYPE)

Step #4: GETCLOBVAL() example

```
select X.OBJECT_VALUE.GETCLOBVAL()
FROM BOOKS_STRUCTURED X;
```

Result:



All Rows Fetched: 1 in 0.141 seconds

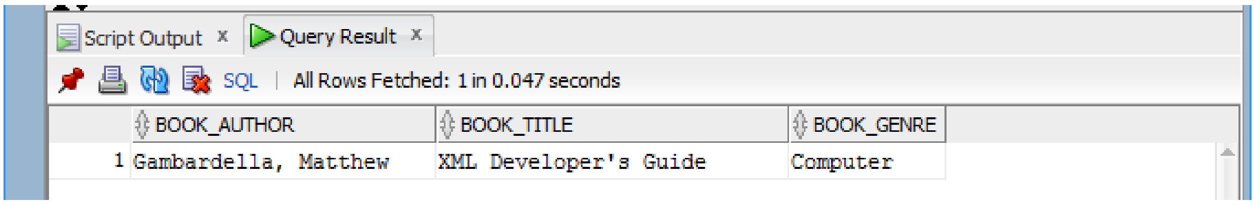
```
  X.OBJECT_VALUE.GETCLOBVAL()
1 <?xml version="1.0" encoding="UTF-8"?><BookCatalog>  <Book id="bk101">    <autho...
```

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:



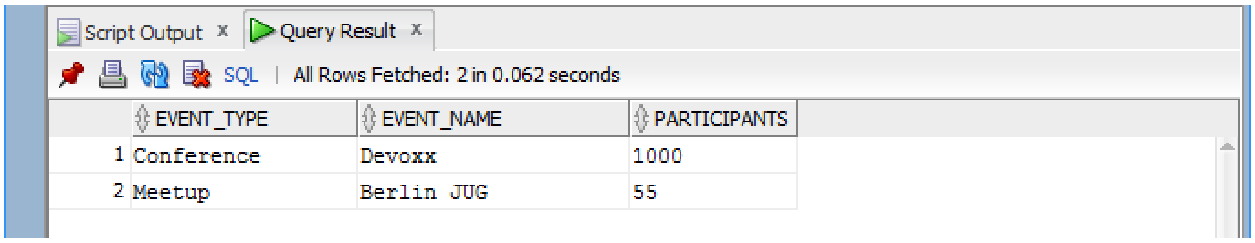
```
  BOOK_AUTHOR            BOOK_TITLE              BOOK_GENRE
1 Gambardella, Matthew   XML Developer's Guide   Computer
```

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:



```
  EVENT_TYPE   EVENT_NAME   PARTICIPANTS
1 Conference   Devoxx       1000
2 Meetup       Berlin JUG   55
```

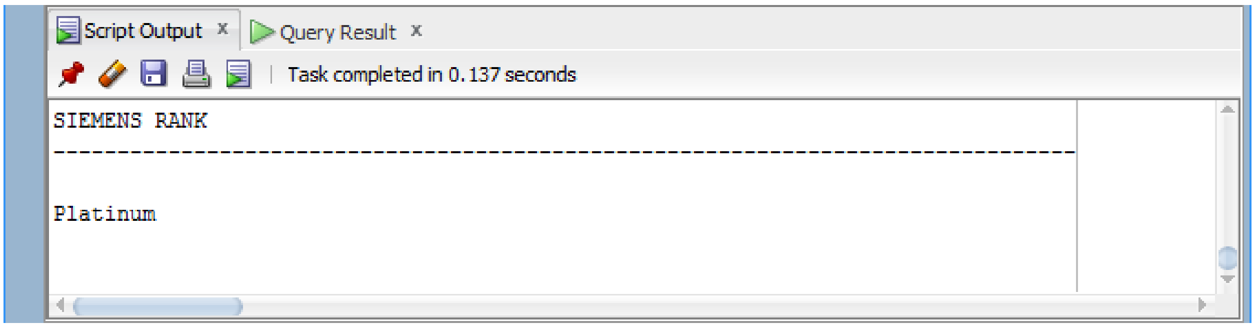
Part #5: XMLQuery examples.

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:



```
SIEMENS RANK
--------------------------------------------------------------------------------

Platinum
```

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:

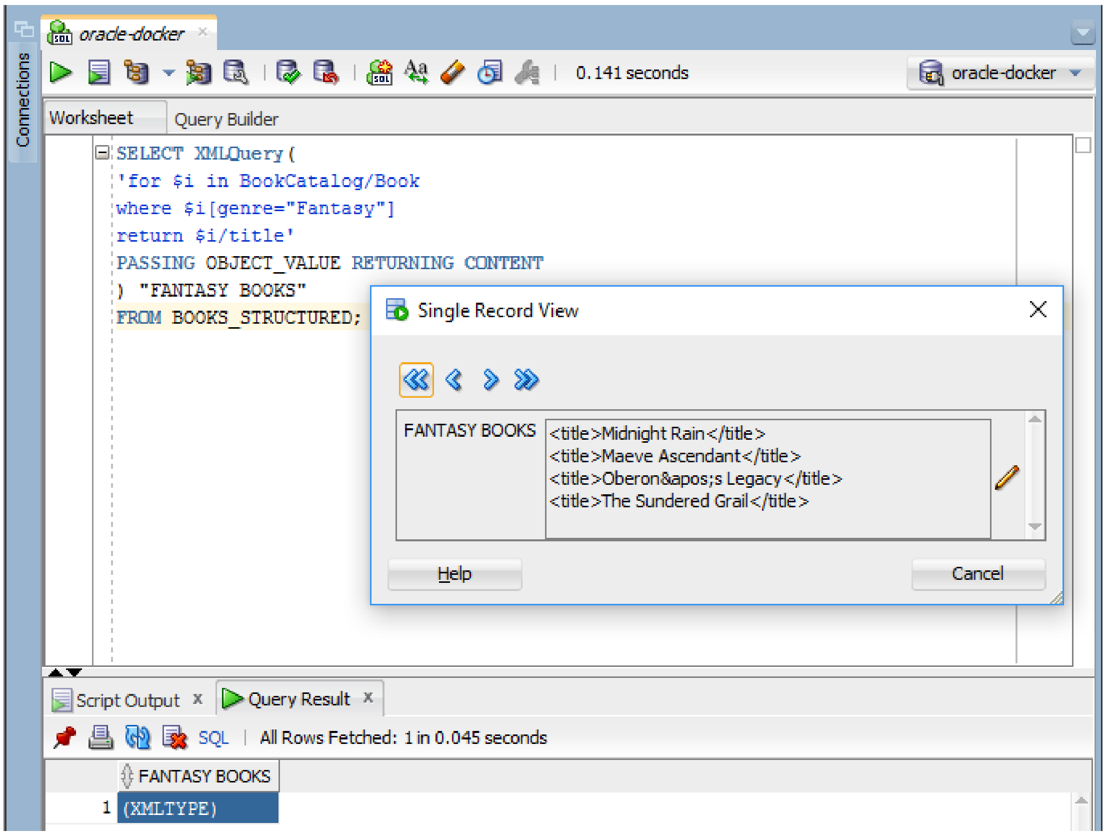
```
BIG EVENTS
--------------------------------------------------------------------
Devoxx
```

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:



```
SELECT XMLQuery(
'for $i in BookCatalog/Book
where $i[genre="Fantasy"]
return $i/title'
PASSING OBJECT_VALUE RETURNING CONTENT
) "FANTASY BOOKS"
FROM BOOKS_STRUCTURED;
```

Single Record View

```
FANTASY BOOKS  <title>Midnight Rain</title>
               <title>Maeve Ascendant</title>
               <title>Oberon&apos;s Legacy</title>
               <title>The Sundered Grail</title>
```

All Rows Fetched: 1 in 0.045 seconds

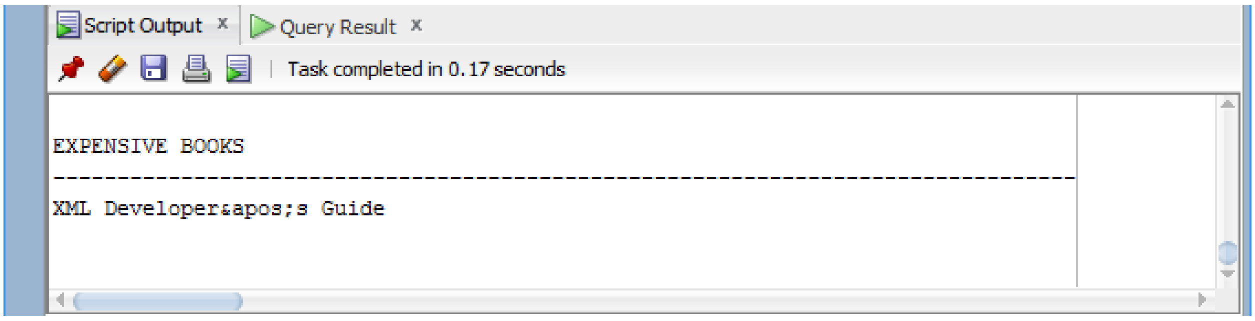
	FANTASY BOOKS
1	(XMLTYPE)

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:



Task completed in 0.17 seconds

```
EXPENSIVE BOOKS
--------------------------------------------------------------------------------
XML Developer&apos;s Guide
```

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

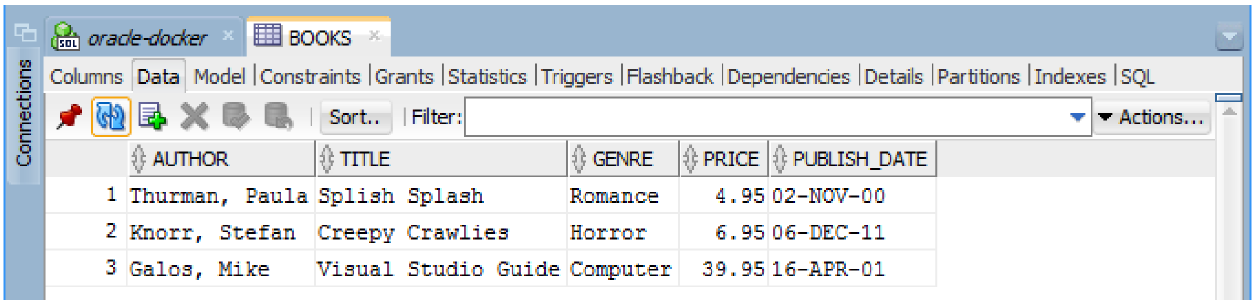
First, I created the table Books

```
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:

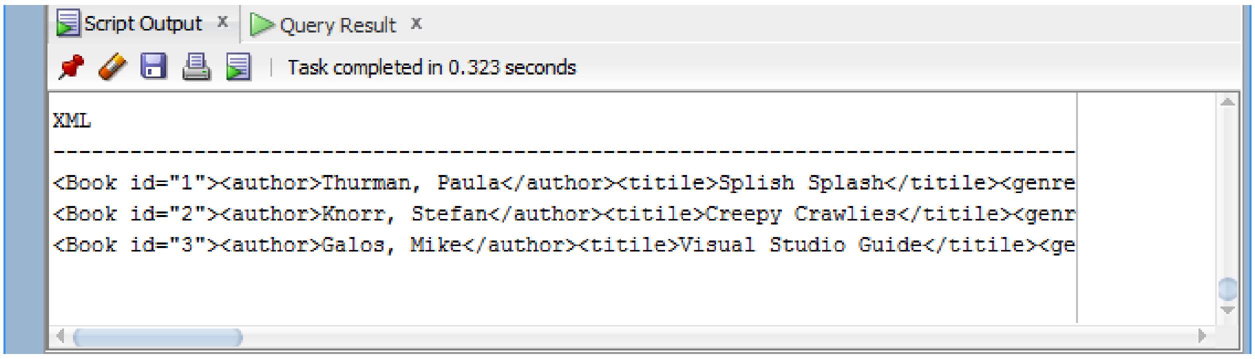


	AUTHOR	TITLE	GENRE	PRICE	PUBLISH_DATE
1	Thurman, Paula	Splish Splash	Romance	4.95	02-NOV-00
2	Knorr, Stefan	Creepy Crawlies	Horror	6.95	06-DEC-11
3	Galos, Mike	Visual Studio Guide	Computer	39.95	16-APR-01

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

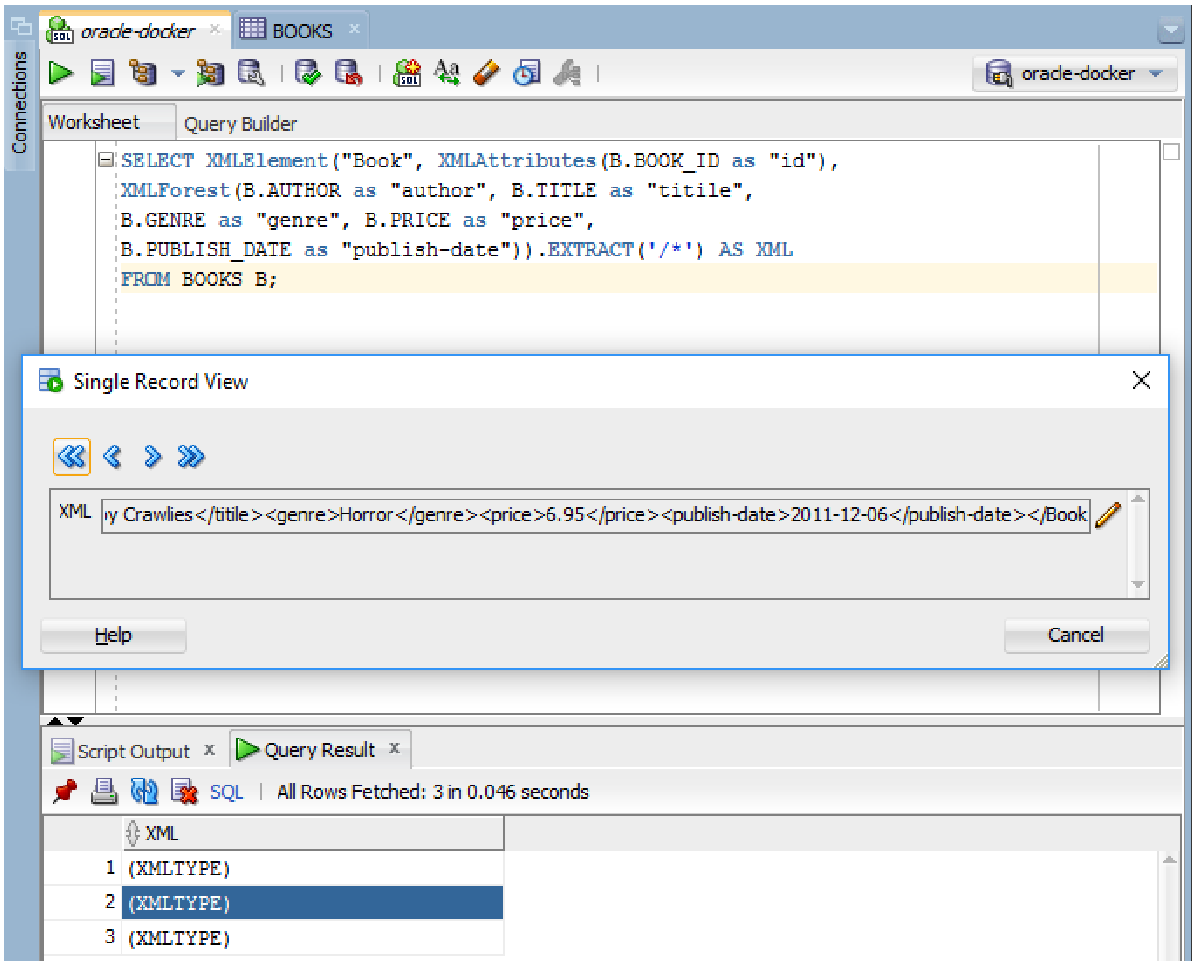
```
SELECT XMLElement("Book", XMLAttributes(B.BOOK_ID as "id"),
XMLForest(B.AUTHOR as "author", B.TITLE as "titile",
B.GENRE as "genre", B.PRICE as "price",
B.PUBLISH_DATE as "publish-date")).EXTRACT('/*') AS XML
FROM BOOKS B;
```

Script output:



```
XML
--------------------------------------------------------------------------------
<Book id="1"><author>Thurman, Paula</author><titile>Splish Splash</titile><genre
<Book id="2"><author>Knorr, Stefan</author><titile>Creepy Crawlies</titile><genr
<Book id="3"><author>Galos, Mike</author><titile>Visual Studio Guide</titile><ge
```

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



```
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;
```

```
XML  ıy Crawlies</titile><genre>Horror</genre><price>6.95</price><publish-date>2011-12-06</publish-date></Book
```

	XML
1	(XMLTYPE)
2	(XMLTYPE)
3	(XMLTYPE)

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.

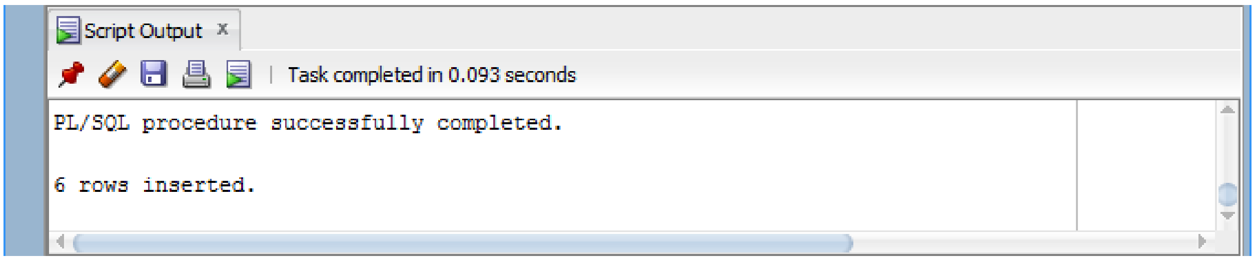
```
CREATE OR REPLACE PROCEDURE LOADBOOKS(FILENAME IN VARCHAR2) AS
books_filebfile;
xml_dataclob;
xml_handleDBMS_XMLSTORE.ctxType;
row_num number;
varxmlType;
csid integer;
dst_offset number :=1;
src_offset number :=1;
lang_ctx number :=dbms_lob.default_lang_ctx;
warning number;
begin
books_file := bfilename('XML_FILES', 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:



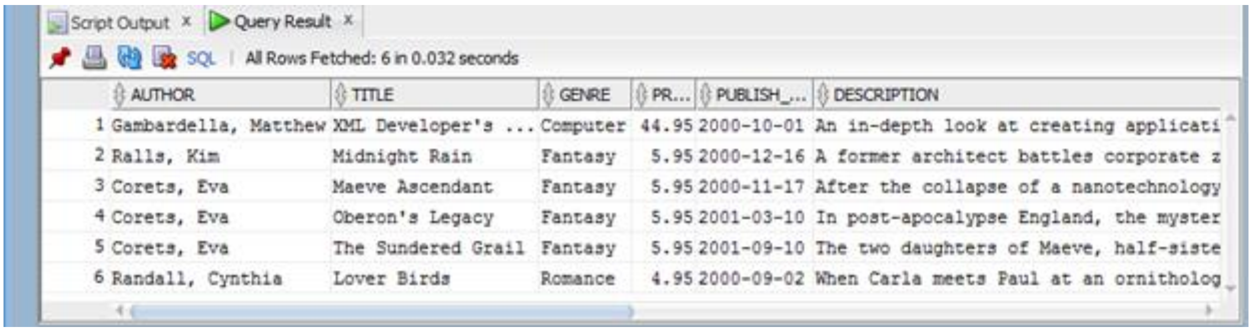
```
PL/SQL procedure successfully completed.

6 rows inserted.
```

Result:

Now I will verify that data was indeed inserted in the table BOOKS_FROM_XML.

SELECT * FROM BOOKS_FROM_XML;



	AUTHOR	TITLE	GENRE	PR...	PUBLISH_...	DESCRIPTION
1	Gambardella, Matthew	XML Developer's ...	Computer	44.95	2000-10-01	An in-depth look at creating applicati
2	Ralls, Kim	Midnight Rain	Fantasy	5.95	2000-12-16	A former architect battles corporate z
3	Corets, Eva	Maeve Ascendant	Fantasy	5.95	2000-11-17	After the collapse of a nanotechnology
4	Corets, Eva	Oberon's Legacy	Fantasy	5.95	2001-03-10	In post-apocalypse England, the myster
5	Corets, Eva	The Sundered Grail	Fantasy	5.95	2001-09-10	The two daughters of Maeve, half-siste
6	Randall, Cynthia	Lover Birds	Romance	4.95	2000-09-02	When Carla meets Paul at an ornitholog

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.