RIGA TECHNICAL UNIVERSITY
FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
INSTITUTE OF APPLIED COMPUTER SYSTEMS

“Technology of Large Databases”
Practical assignment #1
Object-relational DB storage structures

Author: Doston hamrakulov

Studentcardno.: 151ADB089

2016 / 2017study year

Contents

Assignment description:

1. Java class creation and loading in database:
    a. Using LOADJAVA program.
    b. Using CREATE JAVA command.
2. Java class meta-date inspection (SELECT).
3. Creation of PL/SQL function which call Java method. Execution of Java method.
4. Java class creation, loading and testing for SELECT and INSERT queries realization.
5. Conclusions (what seems good, what bad, what like, what is problematic).

1. Java class creation and loading in database:

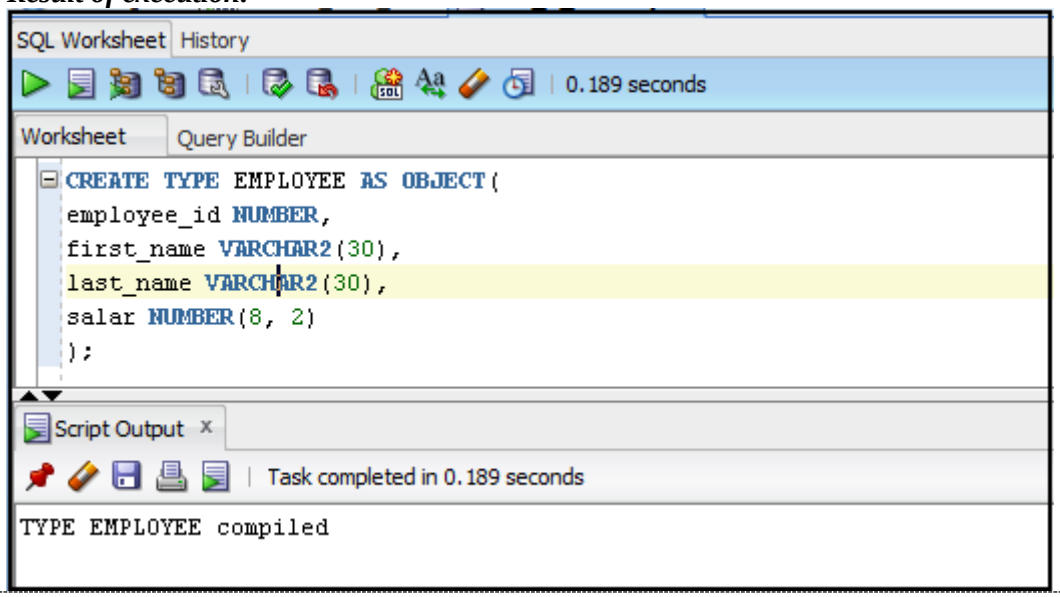
Text in blue and italic: SQL code

Text in red and italic: Java code

Objects in my Database

```
CREATE TYPE EMPLOYEE AS OBJECT(
employee_id NUMBER,
first_name VARCHAR2(30),
last_name VARCHAR2(30),
salar NUMBER(8, 2)
);
```

Result of execution:



I used Eclipse IDE for creating java file but I think that it s not important which IDE we use. Reason why I used Eclipse is that a written code will have its own format like color of script or comment.

1.1 Using LOADJAVA program

1.1.1 Define a class, `Student`, as follows:

```
public class Student {


	public static String myStr() {

		//Creating array from objects
		String student[] = new String[5];


		String myString = "";

		//Filling array based on Object Construction
		student[0] = "Doston";
		student[1] = "Hamrakulov";
```

```
            student[2] = "$1000000";
            student[3] = "Samarkand";
            student[4] = "Uzbekistan";

            for (int i = 0; i < student.length; i++) {
                  myString = myString + student[i] + "   ";
            }
            return myString;
      }

      public static void main(String[] args) {
            Student st = new Student();
            System.out.println(st.myStr());
      }
}
```

1.1.2 Compile the class on your client system using the standard Java compiler, as follows

>javac Student.java

Result of execution:

```
C:\Java_files_for_database>javac Student.java

C:\Java_files_for_database>java Student
Doston   Hamrakulov   $1000000   Samarkand   Uzbekistan

C:\Java_files_for_database>
```

1.1.3 Load the class on the server using `loadjava`. You must specify the user name and password. Run the `loadjava` command as follows:

loadjava -u HR/hr@localhost:1521/pdborcl -v -r -t Student.java

Result of execution:

Command Prompt

```
C:\Java_files_for_database>loadjava -u HR/hr@localhost:1521/pdborcl -v -r -t Student.java
arguments: '-u' 'HR/***@localhost:1521/pdborcl' '-v' '-r' '-t' 'Student.java'
creating : source Student
loading  : source Student
resolving: source Student
Classes Loaded: 0
Resources Loaded: 0
Sources Loaded: 1
Published Interfaces: 0
Classes generated: 0
Classes skipped: 0
Synonyms Created: 0
Errors: 0

C:\Java_files_for_database>
```

1.1.4 Publish the stored procedure through a call specification

In SQL Developer, connect to the database and define a top-level call specification for `Student.myStr()` as follows

```
CREATE OR REPLACE FUNCTION Student RETURN VARCHAR2 AS
  LANGUAGE JAVA NAME 'Student.myStr () return java.lang.String';
/
```

Result of execution:

```
SQL> CREATE OR REPLACE FUNCTION Student RETURN VARCHAR2 AS
  2    LANGUAGE JAVA NAME 'Student.myStr () return java.lang.String';
  3  /

Function created.

SQL>
```

1.1.5 Call the stored procedure, as follows:

```
VARIABLE myString VARCHAR2(50);
CALL Student() INTO :myString;

PRINT myString;
```

Result of execution:

```
SQL> VARIABLE myString VARCHAR2(50);
SQL> CALL Student() INTO :myString;

Call completed.

SQL> PRINT myString;

MYSTRING
--------------------------------------------------------------------------------
Doston  Hamrakulov  $1000000  Samarkand  Uzbekista

SQL>
```

1.2 Using CREATE JAVA command.

1.2.1 Create Java a file named Billionaire in SQL Developer using CREATE JAVA

Use the `CREATE JAVA` statement to create a schema object containing a Java source, class, or resource.

```
CREATE or replace JAVA source named "Billionaire" AS
  public class Billionaire {
```

```
	public static String myStr() {

		//Creating array from objects
		String billionaire[] = new String[5];


		String myString = "";

		//Filling array based on Object Construction
		billionaire[0] = "Doston";
		billionaire[1] = "Hamrakulov";
		billionaire[2] = "$1 000 000 000 000 000m";
		billionaire[3] = "Samarkand";
		billionaire[4] = "Uzbekistan";

		for (int i = 0; i < billionaire.length; i++) {
			myString = myString + billionaire[i] + "  ";
		}
		return myString;
	}

	public static void main(String[] args) {
		Billionaire st = new Billionaire();
		System.out.println(st.myStr());
	}
}
/
```

Result of execution:

```
Java created.

SQL>
```

1.2.2 Publish the stored procedure through a call specification

In SQL Developer, connect to the database and define a top-level call specification for `Billionaire.myStr()` as follows

```
CREATE OR REPLACE FUNCTION Billionaire RETURN VARCHAR2 AS
 LANGUAGE JAVA NAME 'Billionaire.myStr () return java.lang.String';
/
```

Result of execution:

```
SQL> CREATE OR REPLACE FUNCTION Billionaire RETURN VARCHAR2 AS
  2    LANGUAGE JAVA NAME 'Billionaire.myStr () return java.lang.String';
  3  /

Function created.

SQL>
```

1.1.5 Call the stored procedure, as follows:

```
VARIABLE myString VARCHAR2(50);
CALL Student() INTO :myString;

PRINT myString;
```

Result of execution:

```
SQL> VARIABLE myString VARCHAR2(50);
SQL> CALL Student() INTO :myString;

Call completed.

SQL> PRINT myString;

MYSTRING
--------------------------------------------------------------------------------
Doston  Hamrakulov  $1000000  Samarkand  Uzbekista

SQL>
English (United States)
```

Result of execution:

Result of execution:

Result of execution:

Result of execution:

Result of execution:

Result of execution:

Result of execution:

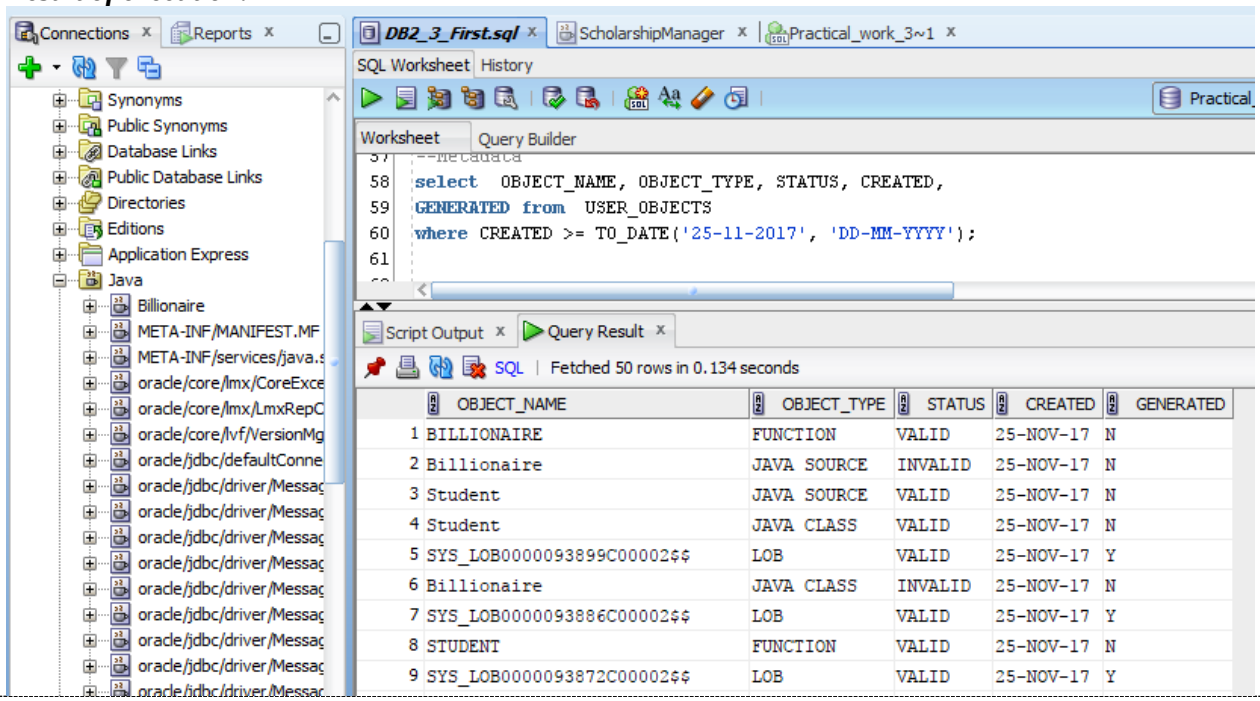
Result of execution:

Result of execution:

1.3 Metadata

select OBJECT_NAME, OBJECT_TYPE, STATUS, CREATED,
GENERATED from USER_OBJECTS
where CREATED >= TO_DATE('25-11-2017', 'DD-MM-YYYY');

Result of execution:



2. Java class meta-date inspection (SELECT).

3. Java class methods of execution SQL command (data output)

3.1 About planning database schema

3.2 Creating database tables

After planning the database schema, create the database tables required by the schema plan. I will created 4 tables in sequence order:

```
CREATE TABLE University (
Univ_id NUMBER(3) NOT NULL,
Title VARCHAR2(30) NOT NULL,
Street VARCHAR2(20) NOT NULL,
City VARCHAR2(20) NOT NULL,
State CHAR(2) NOT NULL,
Zip VARCHAR2(10) NOT NULL,
Phone VARCHAR2(12),
PRIMARY KEY (Univ_id)
);
/
```

Result of execution:

```
SQL>
SQL> CREATE TABLE University (
  2  Univ_id NUMBER(3) NOT NULL,
  3  Title VARCHAR2(30) NOT NULL,
  4  Street VARCHAR2(20) NOT NULL,
  5  City VARCHAR2(20) NOT NULL,
  6  State CHAR(2) NOT NULL,
  7  Zip VARCHAR2(10) NOT NULL,
  8  Phone VARCHAR2(12),
  9  PRIMARY KEY (Univ_id)
 10  );

Table created.

SQL> /
```

```
CREATE TABLE Applicant (
Applicant_id NUMBER(3) NOT NULL,
Name VARCHAR2(30) NOT NULL,
Surname VARCHAR2(30) NOT NULL,
Country VARCHAR2(30) NOT NULL,
University NUMBER(3) REFERENCES University,
PRIMARY KEY (Applicant_id)
);
/
```

Result of execution:

```
SQL> CREATE TABLE Applicant (
  2  Applicant_id NUMBER(3) NOT NULL,
  3  Name VARCHAR2(30) NOT NULL,
  4  Surname VARCHAR2(30) NOT NULL,
  5  Country VARCHAR2(30) NOT NULL,
  6  University NUMBER(3) REFERENCES University,
  7  PRIMARY KEY (Applicant_id)
  8  );

Table created.

SQL> /
```

```
CREATE TABLE Scholarship (
Scholar_id NUMBER(4) PRIMARY KEY,
Description VARCHAR2(20),
Duration VARCHAR2(30),
Grant_amount NUMBER(3)
);
/
```

Result of execution:

```
SQL> CREATE TABLE Scholarship (
  2  Scholar_id NUMBER(4) PRIMARY KEY,
  3  Description VARCHAR2(20),
  4  Duration VARCHAR2(30),
  5  Grant_amount NUMBER(3)
  6  );

Table created.

SQL>
```

```
CREATE TABLE LineApplications (
LineNo NUMBER(2),
Applicant_id NUMBER(3) REFERENCES Applicant,
Scholar_id NUMBER(4) REFERENCES Scholarship,
Deadline VARCHAR2(20),
PRIMARY KEY (LineNo, Applicant_id)
);
/
```

Result of execution:

```
SQL> CREATE TABLE LineApplications (
  2  LineNo NUMBER(2),
  3  Applicant_id NUMBER(3) REFERENCES Applicant,
  4  Scholar_id NUMBER(4) REFERENCES Scholarship,
  5  Deadline VARCHAR2(20),
  6  PRIMARY KEY (LineNo, Applicant_id)
  7  );

Table created.

SQL>
```

Result of execution:

3.3 Writing the Java classes

After creating the database tables, I consider the operations required in a scholarship applications database system and write the appropriate Java methods. In a simple system based on the tables defined in the preceding examples, I need methods for registering customers, stocking parts, entering orders, and so on. I can implement these methods in a Java class, `ScholarshipManager`, as follows:

```
import java.sql.*;
import java.io.*;
import oracle.jdbc.*;

public class ScholarshipManager
{
      public static void addUniversity (int univer_id, String Title, String street,
                        String city, String state, String zipCode, String phoneNo)
throws SQLException
                      {
                        String sql = "INSERT INTO Customers VALUES (?,?,?,?,?,?,?)";
                        try
                        {
                          Connection conn =
DriverManager.getConnection("jdbc:default:connection:");
                          PreparedStatement pstmt = conn.prepareStatement(sql);
                          pstmt.setInt(1, univer_id);
                          pstmt.setString(2, Title);
                          pstmt.setString(3, street);
                          pstmt.setString(4, city);
                          pstmt.setString(5, state);
                          pstmt.setString(6, zipCode);
                          pstmt.setString(7, phoneNo);
                          pstmt.executeUpdate();
                          pstmt.close();
                        }
                        catch (SQLException e)
                        {
                          System.err.println(e.getMessage());
                        }
                      }

  public static void addScholarship (int scholar_id, String description, String
duration, int grant_amount)
                                                                  throws SQLException
  {
    String sql = "INSERT INTO StockItems VALUES (?,?,?,?)";
    try
    {
      Connection conn = DriverManager.getConnection("jdbc:default:connection:");
      PreparedStatement pstmt = conn.prepareStatement(sql);
      pstmt.setInt(1, scholar_id);
      pstmt.setString(2, description);
      pstmt.setString(3, duration);
```

```
      pstmt.setInt(4, grant_amount);
      pstmt.executeUpdate();
      pstmt.close();
    }
    catch (SQLException e)
    {
      System.err.println(e.getMessage());
    }
  }


  public static void enterApplicant (int applicant_id, String name, String surname,
String country, int university) throws SQLException
  {
    String sql = "INSERT INTO Customers VALUES (?,?,?,?,?)";
    try
    {
      Connection conn = DriverManager.getConnection("jdbc:default:connection:");
      PreparedStatement pstmt = conn.prepareStatement(sql);
      pstmt.setInt(1, applicant_id);
      pstmt.setString(2, name);
      pstmt.setString(3, surname);
      pstmt.setString(4, country);
      pstmt.setInt(5, university);
      pstmt.executeUpdate();
      pstmt.close();
    }
    catch (SQLException e)
    {
      System.err.println(e.getMessage());
    }
  }

  public static void addLineApplication (int lineNo, int applicant_id, int
scholar_id, String deadline) throws SQLException
  {
    String sql = "INSERT INTO LineItems VALUES (?,?,?,?)";
    try
    {
      Connection conn = DriverManager.getConnection("jdbc:default:connection:");
      PreparedStatement pstmt = conn.prepareStatement(sql);
      pstmt.setInt(1, lineNo);
      pstmt.setInt(2, applicant_id);
      pstmt.setInt(3, scholar_id);
      pstmt.setString(4, deadline);
      pstmt.executeUpdate();
      pstmt.close();
    }
    catch (SQLException e)
    {
      System.err.println(e.getMessage());
    }
  }

  public static void totalApplications () throws SQLException
  {

    String sql = "SELECT A.Applicant_id, ROUND(SUM(S.Grant_amount * S.Grant_amount))
AS TOTAL " +
             "FROM Applicant A, LineApplications L, Scholarship S " +
             "WHERE A.Applicant_id = L.Applicant_id AND L.Scholar_id = S.Scholar_id "
+
```

```
                "GROUP BY A.Applicant_id";
    try
    {
      Connection conn = DriverManager.getConnection("jdbc:default:connection:");
      PreparedStatement pstmt = conn.prepareStatement(sql);
      ResultSet rset = pstmt.executeQuery();
      int count = 0;

      while (rset.next()) {
          ++count;

      }

      if (count == 0) {
          System.out.println("No data found");
      }
      System.out.println(count);
      rset.close();
      pstmt.close();
    }
    catch (SQLException e)
    {
      System.err.println(e.getMessage());
    }
  }

  public static void deleteApplication (int applicantNo) throws SQLException
  {
    String sql = "DELETE FROM LineApplications WHERE Applicant_id = ?";
    try
    {
      Connection conn = DriverManager.getConnection("jdbc:default:connection:");
      PreparedStatement pstmt = conn.prepareStatement(sql);
      pstmt.setInt(1, applicantNo);
      pstmt.executeUpdate();
      sql = "DELETE FROM Applicant WHERE Applicant_id = ?";
      pstmt = conn.prepareStatement(sql);
      pstmt.setInt(1, applicantNo);
      pstmt.executeUpdate();
      pstmt.close();
    }
    catch (SQLException e)
    {
      System.err.println(e.getMessage());
    }
  }
}
```

Result of execution:

3.4 Loading the Java classes

After writing the Java classes, use the `loadjava` tool to upload my Java stored procedures into Oracle Database, as follows:

```
loadjava -u HR/hr@localhost:1521/pdborcl -v -r -t Scholarshipmanager.java
```

Result of execution:

```
Command Prompt

C:\Users\dosto>cd ../..

C:\>cd Java_files_for_database

C:\Java_files_for_database>loadjava -u HR/hr@localhost:1521/pdborcl -v -r -t Scholarshipmanager.java
arguments: '-u' 'HR/***@localhost:1521/pdborcl' '-v' '-r' '-t' 'Scholarshipmanager.java'
creating : source ScholarshipManager
loading  : source ScholarshipManager
created  : CREATE$JAVA$LOB$TABLE
resolving: source ScholarshipManager
Classes Loaded: 0
Resources Loaded: 0
Sources Loaded: 1
Published Interfaces: 0
Classes generated: 0
Classes skipped: 0
Synonyms Created: 0
Errors: 0

C:\Java_files_for_database>
```

3.5 Publishing the Java classes

After loading the Java classes, publish my Java stored procedures in the Oracle data dictionary. To do this, I must write call specifications that map Java method names, parameter types, and return types to their SQL counterparts.

The methods in the `ScholarshipManager` Java class are logically related. I can group their call specifications in a PL/SQL package. To do this, first, let`s create the package specification, as follows:

```
CREATE OR REPLACE PACKAGE scholar_mgr AUTHID CURRENT_USER AS
PROCEDURE add_university (univ_id NUMBER, title VARCHAR2,
street VARCHAR2, city VARCHAR2, state CHAR, zip VARCHAR2, phone VARCHAR2);
PROCEDURE add_scholarship (scholar_id NUMBER, description VARCHAR2,
duration VARCHAR2, grant_amount NUMBER);
PROCEDURE enter_applicant (applicant_id NUMBER, name VARCHAR2,
surname VARCHAR2, country VARCHAR2, university NUMBER);
PROCEDURE add_line_application( lineNo NUMBER, applicant_id NUMBER,
scholar_id NUMBER, deadline VARCHAR);
PROCEDURE total_applications;
PROCEDURE delete_application (applicant_id NUMBER);
END scholar_mgr;
/
```

Result of execution:

```
SQL>
SQL> CREATE OR REPLACE PACKAGE scholar_mgr AUTHID CURRENT_USER AS
  2  PROCEDURE add_university (univ_id NUMBER, title VARCHAR2,
  3  street VARCHAR2, city VARCHAR2, state CHAR, zip VARCHAR2, phone VARCHAR2);
  4  PROCEDURE add_scholarship (scholar_id NUMBER, description VARCHAR2,
  5  duration VARCHAR2, grant_amount NUMBER);
  6  PROCEDURE enter_applicant (applicant_id NUMBER, name VARCHAR2,
  7  surname VARCHAR2, country VARCHAR2, university NUMBER);
  8  PROCEDURE add_line_application( lineNo NUMBER, applicant_id NUMBER,
  9  scholar_id NUMBER, deadline VARCHAR);
 10  PROCEDURE total_applications;
 11  PROCEDURE delete_application (applicant_id NUMBER);
 12  END scholar_mgr;
 13  /

Package created.

SQL>
```

Then, let`s create the package body by writing call specifications for the Java methods, as follows:

```
CREATE OR REPLACE PACKAGE BODY scholar_mgr AS
--University
PROCEDURE add_university (univ_id NUMBER, title VARCHAR2,
street VARCHAR2, city VARCHAR2, state CHAR, zip VARCHAR2, phone VARCHAR2) AS LANGUAGE JAVA
NAME 'ScholarshipManager.addUniversity(int, java.lang.String, java.lang.String,
                                       java.lang.String, java.lang.String, java.lang.String, java.lang.String)';
--Scholarship
PROCEDURE add_scholarship (scholar_id NUMBER, description VARCHAR2,
duration VARCHAR2, grant_amount NUMBER) AS LANGUAGE JAVA
NAME 'ScholarshipManager.addScholarship(int, java.lang.String, java.lang.String, int)';

--Applicant
PROCEDURE enter_applicant (applicant_id NUMBER, name VARCHAR2,
surname VARCHAR2, country VARCHAR2, university NUMBER) AS LANGUAGE JAVA
NAME 'ScholarshipManager.enterApplicant(int, java.lang.String, java.lang.String, java.lang.String, int)';

--LineApplications
PROCEDURE add_line_application(lineNo NUMBER, applicant_id NUMBER,
scholar_id NUMBER, deadline VARCHAR) AS LANGUAGE JAVA
NAME 'ScholarshipManager.addLineApplication(int, int, int, java.lang.String)';

--total_application for totalApplication method in Java
PROCEDURE total_applications AS LANGUAGE JAVA
NAME 'ScholarshipManager.totalApplications()';

--for delete application method in Java
PROCEDURE delete_application (applicant_id NUMBER) AS LANGUAGE JAVA
NAME 'ScholarshipManager.deleteApplication(int)';
END scholar_mgr;
```

Result of execution:

```
  --total_application for totalApplication method in Java
  PROCEDURE total_applications AS LANGUAGE JAVA
  NAME 'ScholarshipManager.totalApplications()';

  --for delete application method in Java
  PROCEDURE delete_application (applicant_id NUMBER) AS LANGUAGE JAVA
  NAME 'ScholarshipManager.deleteApplication(int)';
  END scholar_mgr;
  /

Package body created.

SQL>
```

3.6 Calling the Java stored Procedure

After publishing the Java classes, call my Java stored procedures from the top level and from database triggers, SQL data manipulation language (DML) statements, and PL/SQL blocks.

From an anonymous PL/SQL block, I may start applying for new scholarship in the system by implementing parts, as follows:

```
BEGIN
 scholar_mgr.add_scholarship(2010, 'New UzbekGrant', '6 month', 900);
 scholar_mgr.add_scholarship(2011, 'Young UzbekGrant', '6 month', 800);
 scholar_mgr.add_scholarship(2012, 'New UzbekGrant', '6 month', 900);
 scholar_mgr.add_scholarship(2013, 'Korean Grant', '6 month', 900);
 scholar_mgr.add_scholarship(2014, 'German Grant', '6 month', 100);
 scholar_mgr.add_scholarship(2015, 'Samarkand Grant', '6 month', 300);
 scholar_mgr.add_scholarship(2016, 'Latvian Grant', '2 month', 500);
 COMMIT;
END;
```

Result of execution:

```
SQL>
SQL> BEGIN
  2    scholar_mgr.add_scholarship(2010, 'New UzbekGrant', '6 month', 900);
  3    scholar_mgr.add_scholarship(2011, 'Young UzbekGrant', '6 month', 800);
  4    scholar_mgr.add_scholarship(2012, 'New UzbekGrant', '6 month', 900);
  5    scholar_mgr.add_scholarship(2013, 'Korean Grant', '6 month', 900);
  6    scholar_mgr.add_scholarship(2014, 'German Grant', '6 month', 100);
  7    scholar_mgr.add_scholarship(2015, 'Samarkand Grant', '6 month', 300);
  8    scholar_mgr.add_scholarship(2016, 'Latvian Grant', '2 month', 500);
  9    COMMIT;
 10  END;
 11  /

PL/SQL procedure successfully completed.

SQL>
```

```
BEGIN
 scholar_mgr.add_university(100, 'Uzbek National Uni', 'Beruniy 12', 'Tashkent', 'UZ', '07000',
'+99893777703');
 scholar_mgr.add_university(101, 'Westmenester', 'Nurota 34', 'Samarkand', 'UZ', '04450',
'+99893777703');
 scholar_mgr.add_university(102, 'RTU', 'KALKU 1', 'RIGA', 'LV', '07000', '+371232323');
 COMMIT;
END;
```

Result of execution:

```
SQL> BEGIN
  2    scholar_mgr.add_university(100, 'Uzbek National Uni', 'Beruniy 12', 'Tashkent', 'UZ', '07000', '+99893777703');
  3    scholar_mgr.add_university(101, 'Westmenester', 'Nurota 34', 'Samarkand', 'UZ', '04450', '+99893777703');
  4    scholar_mgr.add_university(102, 'RTU', 'KALKU 1', 'RIGA', 'LV', '07000', '+371232323');
  5    COMMIT;
  6  END;
  7  /

PL/SQL procedure successfully completed.

SQL>
```

```
BEGIN
 scholar_mgr.enter_applicant(500, 'Doston', 'Hamrakulov', 'Uzbekistan', 100);
 scholar_mgr.add_line_application(01, 500, 2010, '21-SEP-2019');
 scholar_mgr.add_line_application(02, 500, 2012, '21-SEP-2019');
 scholar_mgr.add_line_application(03, 500, 2015, '21-SEP-2019');

 scholar_mgr.enter_applicant(501, 'Orif', 'Doniyarov', 'Uzbekistan', 101);
 scholar_mgr.add_line_application(04, 501, 2012, '21-SEP-2019');
 scholar_mgr.add_line_application(05, 501, 2013, '21-DEC-2019');
 scholar_mgr.add_line_application(06, 501, 2016, '21-OCT-2019');

 scholar_mgr.enter_applicant(502, 'John', 'Smith', 'UK', 102);
 scholar_mgr.add_line_application(07, 502, 2012, '21-JUN-2018');
 scholar_mgr.add_line_application(08, 502, 2011, '21-JUN-2018');
 scholar_mgr.add_line_application(09, 502, 2013, '21-JUN-2018');
 COMMIT;
END;
```

Result of execution:

```
SQL>
SQL> BEGIN
  2    scholar_mgr.enter_applicant(500, 'Doston', 'Hamrakulov', 'Uzbekistan', 100);
  3    scholar_mgr.add_line_application(01, 500, 2010, '21-SEP-2019');
  4    scholar_mgr.add_line_application(02, 500, 2012, '21-SEP-2019');
  5    scholar_mgr.add_line_application(03, 500, 2015, '21-SEP-2019');
  6
  7    scholar_mgr.enter_applicant(501, 'Orif', 'Doniyarov', 'Uzbekistan', 101);
  8    scholar_mgr.add_line_application(04, 501, 2012, '21-SEP-2019');
  9    scholar_mgr.add_line_application(05, 501, 2013, '21-DEC-2019');
 10    scholar_mgr.add_line_application(06, 501, 2016, '21-OCT-2019');
 11
 12    scholar_mgr.enter_applicant(502, 'John', 'Smith', 'UK', 102);
 13    scholar_mgr.add_line_application(07, 502, 2012, '21-JUN-2018');
 14    scholar_mgr.add_line_application(08, 502, 2011, '21-JUN-2018');
 15    scholar_mgr.add_line_application(09, 502, 2013, '21-JUN-2018');
 16    COMMIT;
 17  END;
 18  /

PL/SQL procedure successfully completed.

SQL>
```

```
SET SERVEROUTPUT ON
CALL dbms_java.set_output(2000);

CALL scholar_mgr.total_applications();
```

Result of execution:

```
SQL> CALL scholar_mgr.total_applications();
9

Call completed.

SQL>
```

Result of execution:

Result of execution:

Result of execution:

Result of execution:

Result of execution:

Result of execution:

Result of execution:

Result of execution:

Result of execution:

Conclusions.

In this first practical work of the course, was viewed object-relational database principle to form a library database. The paper was made both object tables and relational tables with object columns and tables with a collection of objects.

Even though I used to work in MySQL, for this practical work or course I have been learning Oracle and Pl/SQL to implement all required tasks in the practical work.

Every task had new things for me to learn and of course it had also some error or difficulties to combine them. But I am totally sure that that every task pushed me forward and increased my interest to learn deeper Oracle and Pl/SQL skills.

To sum up, I have gained a lot of skill of Oracle and Pl/SQL and certainly there are more which I should learn. Therefore, I think I will improve my knowledge till the ending of semester by doing next tasks and in lectures.

LineApplications			
LineNo NUMBER PK	Applicant_id NUMBER PK, FK	Scholar_id NUMBER FK	Deadline VARCHAR

Scholarship			
Scholar_id NUMBER PK	Description VARCHAR	Duration VARCHAR	Deadline VARCHAR

Applicant				
Applicant_id NUMBER PK	University NUMBER FK	Name VARCHAR	Surname VARCHAR	Country VARCHAR

University						
Univ_id NUMBER PK	Title VARCHAR	Street VARCHAR	City VARCHAR	State VARCHAR	Zipcode VARCHAR	Phone VARCHAR