

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: 4](#_Toc497481483)

[1. Creation of object table with row type objects. 5](#_Toc497481484)

[Creation 5](#_Toc497481485)

[Data Input 6](#_Toc497481486)

[Select 6](#_Toc497481487)

[Metadata 7](#_Toc497481488)

[Creating a Member Method 8](#_Toc497481489)

[2. Creation of object column and object table. 9](#_Toc497481490)

[Creation 9](#_Toc497481491)

[Data Input 11](#_Toc497481492)

[Select 12](#_Toc497481493)

[Metadata 12](#_Toc497481494)

[3. Creation of object view 13](#_Toc497481495)

[Creation 13](#_Toc497481496)

[Data Input 16](#_Toc497481497)

[Select 16](#_Toc497481498)

[Metadata 17](#_Toc497481499)

[4. Creation of table with collection. 17](#_Toc497481500)

[Creation 17](#_Toc497481501)

[Data Input 19](#_Toc497481502)

[Select 20](#_Toc497481503)

[Metadata 20](#_Toc497481504)

[5. Creation of object type. 21](#_Toc497481505)

[Creation 21](#_Toc497481506)

[Data input 21](#_Toc497481507)

[Select 22](#_Toc497481508)

[Function TREAT() 23](#_Toc497481509)

[Function IS OF TYPE() 25](#_Toc497481510)

[Function ISYS\_TYPEID() 26](#_Toc497481511)

[Metadata 26](#_Toc497481512)

[6. Conclusions. 27](#_Toc497481513)

# Assignment description:

1. Creation of object table with row type objects. Input of data (INSERT). Output of metadata (SELECT), output of objects and its components (SELECT) using function Value(). Map or ORDER and MEMBER methods creation for row type objects. Queries with methods (min. 3).

2. Creation of one to many (1 : N) object relationship between relational table with object column and object table. Input of data. Input of object identifiers using function REF(). Data extraction (all objects and object components) using function DEREF().

3. Creation of object view with 2 methods from 2 tables (relational table and relational table with column objects). Realization of queries from object view (min. 2).

4. Creation of table with collection. For collection objects is created Member type method. Data input. Queries with method (min.2).

5. Creation of object type hierarchy (min. 4 object types) with methods. Creation of object table with different type of objects (according defined type hierarchy). Data input, queries with TREAT(), IS OF TYPE(), SYS\_TTYPEID(). Realization of methods inheritance. Queries.  
6. Conclusions.

# Creation of object table with row type objects.

## Creation

|  |
| --- |
| CREATE TYPE employee\_type AS OBJECT(  employee\_id NUMBER,  first\_name VARCHAR2(30),  last\_name VARCHAR2(30),  email VARCHAR2(30),  hire\_date DATE,  job\_id varchar2(30),  salar NUMBER(8, 2),  department VARCHAR2(30)  ); |
| Result of execution: |

|  |
| --- |
| CREATE TABLE employee of employee\_type;  ALTER TABLE employee  ADD PRIMARY KEY (employee\_id); |
| Result of execution: |

## Data Input

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

## Select

|  |
| --- |
| SELECT \* FROM employee; |
| Result of execution: |

|  |
| --- |
| SELECT VALUE(A) FROM employee A; |
| Result of execution: |

We only see the “VALUE(A)”, whereas a double click over the value we want to see will reveal the information.

|  |
| --- |
| Result of execution: |

|  |
| --- |
| SELECT VALUE(A) FROM employee A WHERE value(A).last\_name = 'Hamrakulov'; |
| Result of execution: |

## Metadata

|  |
| --- |
| In SQL Developer, what we do is to go to the left panel, and in CONNECTION>TABLES>Departments and double-click over it, so we can see the metadata of our DB. |
| Result of execution: |

## Creating a Member Method

|  |
| --- |
| I have decided to use other database for creating a Member Methods to simply the implementation and I have created three functions (methods) for empty pool and these functions will calculate the special operations. |
| CREATE OR REPLACE TYPE empty\_pool\_typ AS OBJECT (  length INTEGER,  width INTEGER,  height INTEGER,  MEMBER FUNCTION surface RETURN INTEGER,  MEMBER FUNCTION volume RETURN INTEGER,  MEMBER FUNCTION peremeter RETURN INTEGER,  MEMBER PROCEDURE display (SELF IN OUT NOCOPY empty\_pool\_typ));  /  CREATE OR REPLACE TYPE BODY empty\_pool\_typ AS  MEMBER FUNCTION volume RETURN INTEGER IS  BEGIN  RETURN length \* width \* height;  -- RETURN SELF.len \* SELF.wth \* SELF.hgt; -- equivalent to previous line  END;  MEMBER FUNCTION surface RETURN INTEGER IS  BEGIN -- not necessary to include SELF in following line  RETURN 2 \* (length \* width + length \* height + width \* height);  END;  MEMBER FUNCTION peremeter RETURN INTEGER IS  BEGIN  RETURN length + width + height;  END;  MEMBER PROCEDURE display (SELF IN OUT NOCOPY empty\_pool\_typ) IS  BEGIN  DBMS\_OUTPUT.PUT\_LINE('Length: ' || length || ' - ' || 'Width: ' || width  || ' - ' || 'Height: ' || height);  DBMS\_OUTPUT.PUT\_LINE('Volume: ' || volume || ' - ' || 'Surface area: '  || surface || ' - ' || 'Premeter: ' || peremeter);  END;  END;  /  -- Table creation  CREATE TABLE empty\_pool of empty\_pool\_typ;  -- Insertin data  INSERT INTO empty\_pool VALUES(10, 10, 10);  INSERT INTO empty\_pool VALUES(3, 4, 5);  INSERT INTO empty\_pool VALUES(7, 8, 9);  INSERT INTO empty\_pool VALUES(4, 5, 8);  SELECT \* FROM empty\_pool;  SELECT p.volume(), p.surface(), p.peremeter() FROM empty\_pool p WHERE p.length = 10;  DECLARE  pool empty\_pool\_typ;  BEGIN -- PL/SQL block for selecting a solid and displaying details  SELECT VALUE(p) INTO pool FROM empty\_pool p WHERE p.length = 10;  pool.display();  END;  / |
| Result of execution: |

# Creation of object column and object table.

## Creation

|  |
| --- |
| CREATE OR REPLACE TYPE AddressType AS OBJECT(  street VARCHAR2(15),  city VARCHAR2(15),  state CHAR(2),  zip VARCHAR2(5)  );  /  CREATE Or Replace TYPE PersonType AS OBJECT (  id NUMBER,  first\_name VARCHAR2(10),  last\_name VARCHAR2(10),  dob DATE,  phone VARCHAR2(12),  address AddressType  );  /  CREATE TABLE object\_reader OF PersonType;  INSERT INTO object\_reader VALUES (  PersonType(1, 'Doston', 'Hamrakulov', '04-SEP-1995', '800-555-5555',  AddressType('Kalku 1', 'City', 'LS', '12345')  )  );  INSERT INTO object\_reader VALUES (  PersonType(2, 'John', 'White', '04-FEB-1945', '800-555-5555',  AddressType('2 Ave', 'City', 'UZ', '12345')  )  );  -- Different style of input data  INSERT INTO object\_reader (  id, first\_name, last\_name, dob, phone,  address  ) VALUES (  3, 'Bobosher', 'Hamroyev', '05-FEB-1968', '93-727-08-95',  AddressType('Tashkent 2', 'Town', 'UZ', '12345')  );  INSERT INTO object\_reader (  id, first\_name, last\_name, dob, phone,  address  ) VALUES (  4, 'Rustam', 'Yuldoshev', '13-NOV-1987', '888-888-8888',  AddressType('UBBER 23', 'City', 'MA', '12345')  );  CREATE OR REPLACE TYPE BookType AS OBJECT (  id NUMBER,  name VARCHAR2(15),  description VARCHAR2(22),  price NUMBER(5, 2),  days\_valid NUMBER,  MEMBER FUNCTION getByDate RETURN DATE  );  /  CREATE TABLE object\_book OF BookType;    INSERT INTO object\_book (  id, name, description, price, days\_valid  ) VALUES (  1, 'AAA', 'English Grammar', 2.99, 5  );  INSERT INTO object\_book (  id, name, description, price, days\_valid  ) VALUES (  2, 'BBB', 'Detactive book', 9.99, 10  );  INSERT INTO object\_book (  id, name, description, price, days\_valid  ) VALUES (  3, 'CCC', 'Romantic stories', 9.99, 10  );  /  CREATE TABLE borrow\_book (  id NUMBER PRIMARY KEY,  reader REF PersonType SCOPE IS object\_reader,  book REF BookType SCOPE IS object\_book  );  / |

## Data Input

|  |
| --- |
| INSERT INTO borrow\_book (  id,  reader,  book  ) VALUES (  1,  (SELECT REF(o\_r) FROM object\_reader o\_r WHERE o\_r.id = 1),  (SELECT REF(o\_b) FROM object\_book o\_b WHERE o\_b.id = 1)  );  INSERT INTO borrow\_book (  id,  reader,  book  ) VALUES (  2,  (SELECT REF(o\_r) FROM object\_reader o\_r WHERE o\_r.id = 2),  (SELECT REF(o\_b) FROM object\_book o\_b WHERE o\_b.id = 2)  );  INSERT INTO borrow\_book (  id,  reader,  book  ) VALUES (  3,  (SELECT REF(o\_r) FROM object\_reader o\_r WHERE o\_r.id = 3),  (SELECT REF(o\_b) FROM object\_book o\_b WHERE o\_b.id = 3)  );  / |
| Result of execution: |

## Select

|  |
| --- |
| select \* from borrow\_book; |
| Result of execution |

## Metadata

|  |
| --- |
| In SQL Developer, what we do is to go to the left panel, and in CONNECTION>TABLES>Departments and double-click over it, so we can see the metadata of our DB. |
| Result of execution: |
| There we can notice the object: “ADDRESSTYPE”; |

# Creation of object view

## Creation

**Goal:** Let`s look at how object view might be used in Online Store where online users can allocate their items for selling. Their existing application tracks LAPTOP, DESKTOP and other type of computers that they use when publishing their products. These computers are in the Market, but data about them stored in relational tables. To help the sellers to locate the right computer, each computer has one or more associated keywords, stored in a straight-forward master-detail relationship.

|  |
| --- |
| CREATE TABLE items (  item\_id INTEGER NOT NULL,  item\_name VARCHAR2(512),  item\_type VARCHAR2(12),  price INTEGER,  CONSTRAINT item\_pk PRIMARY KEY (item\_id));  /  Result of execution:    CREATE TABLE keywords (  item\_id INTEGER NOT NULL,  keyword VARCHAR2(45) NOT NULL,  CONSTRAINT keywords\_pk PRIMARY KEY (item\_id, keyword),  CONSTRAINT keywords\_for\_item FOREIGN KEY (item\_id)  REFERENCES items (item\_id));  /  Result of execution:    CREATE TYPE Keyword\_tab\_t AS TABLE OF VARCHAR2(45);  CREATE TYPE items\_t AS OBJECT (  item\_id INTEGER,  item\_name VARCHAR2(512),  item\_t VARCHAR2(12),  price INTEGER,  keywords Keyword\_tab\_t,  MEMBER FUNCTION set\_attributes (new\_item\_name IN VARCHAR2,  new\_item\_type IN VARCHAR2, new\_price IN INTEGER)  RETURN items\_t,  MEMBER FUNCTION set\_keywords (new\_keywords IN Keyword\_tab\_t)  RETURN items\_t,  PRAGMA RESTRICT\_REFERENCES (DEFAULT, RNDS, WNDS, RNPS, WNPS)  );  /  Result of execution:    -- Here is a body  CREATE TYPE BODY items\_t  AS  MEMBER FUNCTION set\_attributes (new\_item\_name IN VARCHAR2,  new\_item\_type IN VARCHAR2, new\_price IN INTEGER)  RETURN items\_t  IS  item\_holder items\_t := SELF;  BEGIN  item\_holder.item\_name := new\_item\_name;  item\_holder.item\_t := new\_item\_type;  item\_holder.price := new\_price;  RETURN item\_holder;  END;  MEMBER FUNCTION set\_keywords (new\_keywords IN Keyword\_tab\_t)  RETURN items\_t  IS  item\_holder items\_t := SELF;  BEGIN  item\_holder.keywords := new\_keywords;  RETURN item\_holder;  END;  END;  /  Result of execution:    CREATE VIEW item\_view  OF items\_t  WITH OBJECT OID (item\_id)  AS  SELECT i.item\_id, i.item\_name, i.item\_type, i.price,  CAST (MULTISET (SELECT keyword  FROM keywords k  WHERE k.item\_id = i.item\_id)  AS Keyword\_tab\_t)  FROM items I;  Result of execution: |

## Data Input

|  |
| --- |
| -- Inserting data  INSERT INTO items VALUES (101, 'Lenova', 'LAPTOP', 813);  INSERT INTO items VALUES (102, 'ASUS', 'LAPTOP', 972);  INSERT INTO KEYWORDS VALUES (101, 'new 8GB');  INSERT INTO KEYWORDS VALUES (101, 'Intel Core i7');  INSERT INTO KEYWORDS VALUES (102, 'used 4GB MEMORY');  INSERT INTO KEYWORDS VALUES (102, 'Intel Intel Core i3');  INSERT INTO KEYWORDS VALUES (102, 'SSD driver, 512GB'); |
| Result of execution: |

## Select

|  |
| --- |
| SELECT item\_id, item\_name, keywords  FROM item\_view; |
| Result of execution:    SELECT item\_id, item\_name, keywords  FROM item\_view i  WHERE i.item\_id =102;  Result of execution: |

## Metadata

# Creation of table with collection.

## Creation

|  |
| --- |
| CREATE TYPE item\_typ AS OBJECT (  idno NUMBER,  name VARCHAR2(30),  description VARCHAR2(20),  MAP MEMBER FUNCTION get\_idno RETURN NUMBER,  MEMBER PROCEDURE display\_details ( SELF IN OUT NOCOPY item\_typ ) );  /  CREATE TYPE BODY item\_typ AS  MAP MEMBER FUNCTION get\_idno RETURN NUMBER IS  BEGIN  RETURN idno;  END;  MEMBER PROCEDURE display\_details ( SELF IN OUT NOCOPY item\_typ ) IS  BEGIN  -- use the put\_line procedure of the DBMS\_OUTPUT package to display details  DBMS\_OUTPUT.put\_line(TO\_CHAR(idno) || ' - ' || name || ' - ' || description);  END;  END;  /  CREATE TYPE computer\_typ AS TABLE OF item\_typ; -- nested table type  /  Result of execution:    - - Using the Constructor Method to Insert Values into a Nested Table:  CREATE TABLE computer\_tab (  group\_no NUMBER,  computer\_column computer\_typ ) -- an instance of nested table  NESTED TABLE computer\_column STORE AS computer\_column\_nt; -- storage table for NT  INSERT INTO computer\_tab VALUES (  100,  computer\_typ( item\_typ(1, 'Lenovo', '8GB Memory'),  item\_typ(2, 'Asus', '4GB MEMORY')));  Result of execution:    **- - The following statement creates an array type***contact\_list\_arr***that has no more than ten elements, each of data type VARCHAR2(8)**  CREATE TYPE email\_list\_arr AS VARRAY(10) OF VARCHAR2(80);  /  -- Creating and Populating a VARRAY Data Type  CREATE TYPE phone\_typ AS OBJECT (  country\_code VARCHAR2(2),  area\_code VARCHAR2(3),  ph\_number VARCHAR2(7));  /  CREATE TYPE phone\_varray\_typ AS VARRAY(5) OF phone\_typ;  /  CREATE TABLE branch\_phone\_list (  brach\_no NUMBER(5),  phone\_list phone\_varray\_typ);  INSERT INTO branch\_phone\_list VALUES (  100,  phone\_varray\_typ( phone\_typ ('01', '650', '5550123'),  phone\_typ ('01', '650', '5550148'),  phone\_typ ('01', '650', '5550192')));  Result of execution:    -- To declare nested table types  CREATE TYPE computer\_typ AS TABLE OF computer\_typ;  -- Creating and Populating Simple Nested Tables  CREATE TABLE laptops (  valid DATE,  feature\_one computer\_typ, -- nested tables (empty)  feature\_two computer\_typ,  feature\_three computer\_typ)  NESTED TABLE feature\_one STORE AS feature\_one\_nt -- storage tables  NESTED TABLE feature\_two STORE AS feature\_two\_nt  NESTED TABLE feature\_three STORE AS feature\_three\_nt;  CREATE INDEX feature\_one\_idno\_idx ON feature\_one\_nt(idno);  CREATE INDEX feature\_two\_idno\_idx ON feature\_two\_nt(idno);  CREATE INDEX feature\_three\_idno\_idx ON feature\_three\_nt(idno);  Result of execution: |

## Data Input

|  |
| --- |
| INSERT INTO laptops (valid) VALUES ('01-JUN-03');  UPDATE laptops  SET feature\_one =  computer\_typ (item\_typ(12, 'ASUS', '2GB memory'),  item\_typ(13, 'ASUS', '4GB memory'),  item\_typ(14, 'ASUS', '8GB memory')),  feature\_two =  computer\_typ (item\_typ(15, 'ACER', '2GB memory'),  item\_typ(16, 'ACER', '6GB memory'),  item\_typ(17, 'ACER', '8GB memory')),  feature\_three =  computer\_typ (item\_typ(18, 'APPLE', '4GB memory'),  item\_typ(19, 'APPLE', '8GB memory'))  WHERE valid = '01-JUN-03'; |
| Result of execution: |

## Select

|  |
| --- |
| SELECT m.idno math\_id, c.idno chem\_id, p.idno physics\_id FROM students s,  TABLE(s.math\_majors) m, TABLE(s.chem\_majors) c, TABLE(s.physics\_majors) p; |
| Result of execution: |

## Metadata

|  |
| --- |
|  |

# Creation of object type.

## Creation

|  |
| --- |
| CREATE OR REPLACE TYPE item AS OBJECT(  item\_id NUMBER,  name VARCHAR2(20),  description VARCHAR2(20)) NOT FINAL;    CREATE OR REPLACE TYPE computer UNDER item(  memory VARCHAR2(20),  valid DATE,  price NUMBER) NOT FINAL;  CREATE OR REPLACE TYPE laptop UNDER computer(  battery\_life varchar2(20)  );  CREATE OR REPLACE TYPE phone UNDER item(  memory VARCHAR2(20),  valid DATE,  price NUMBER  );  CREATE OR REPLACE TYPE tablet UNDER item(  memory VARCHAR2(20),  valid DATE,  price NUMBER  );  CREATE TABLE computers OF computer;  CREATE TABLE items OF item;  Result of execution: |

## Data input

|  |
| --- |
| insert into items values(item('101', 'ASUS', 'touch screen'));  insert into items values(item('102', 'PH', 'SSD card'));  insert into items values (computer('103', 'TOSHIBA', 'SS card', '1024 GB', '03-JUN-17', 555));  insert into items values (computer('104', 'TOSHIBA', 'touch screen', '128 GB', '03-MAY-17', 555));  insert into items values (computer('105', 'APPLE', 'touch screen', '512 GB', '03-SEP-16', 555));  insert into items values (computer('106', 'LENOVO', 'touch screen', '256 GB', '03-AUG-17', 555));  insert into items values (laptop('107', 'TOSHIBA', 'touch screen', '128 GB', '03-MAY-17', 555, '8 HOURS'));  insert into items values (laptop('108', 'APPLE', 'touch screen', '512 GB', '03-SEP-16', 555, '9 hours'));  insert into items values (laptop('109', 'LENOVO', 'touch screen', '256 GB', '03-AUG-17', 555, '10 hours'));  insert into items values (phone('110', 'iphone', '5g internet', '128 GB', '03-MAY-17', 100));  insert into items values (phone('111', 'samsung', 'edge screen', '512 GB', '03-SEP-16', 50));  insert into items values (phone('112', 'nokia', 'dual sim', '256 GB', '03-AUG-17', 5));  insert into items values (tablet('110', 'tablet 1', 'unlimited speed', '128 GB', '03-MAY-17', 145));  insert into items values (tablet('111', 'tablet 2', 'edge screen', '512 GB', '03-SEP-16', 5022));  insert into items values (tablet('112', 'tablet 3', 'unlimited memory', '256 GB', '03-AUG-17', 34));  Result of execution: |

## Select

|  |
| --- |
| select VALUE(A) from items A; |
| Result of execution:    select A.\* from items A;  Result of execution:    select VALUE(A).name, VALUE(A).description  from items A;  Result of execution:   Function TREAT() select TREAT(VALUE(A) as item)  from items A;  Result of execution:    select TREAT(VALUE(A) as laptop)  from items A;  select TREAT(VALUE(A) as laptop).battery\_life as BATTERY  from items A;  Result of execution:    select VALUE(A)  from items A  where TREAT(VALUE(A) as phone).price = 100;  Result of execution:   Function IS OF TYPE() select VALUE(A)  from items A  where VALUE(A) IS OF TYPE (laptop);  Result of execution:    select VALUE(A)  from items A  where VALUE(A) IS OF TYPE (computer, tablet);  Result of execution:    select VALUE(A).name, VALUE(A).description  from items A  where VALUE(A) IS OF TYPE (phone, tablet);  Result of execution:   Function ISYS\_TYPEID() select A.name, SYS\_TYPEID(VALUE(A)) TYPE\_ID  from items A;  Result of execution: |

## Metadata

|  |
| --- |
|  |

# 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.