OBJECT RELATIONAL LAB (1)

Object Types, Row Objects, Column Objects, & Object Tables

Ensure you keep a copy of your working statements for upload. You will also reference back to them through the lab. Exercises are the end of each section.

Make sure to read the section before starting the exercises.

When creating object types, run them a separated statements

1. CREATING OBJECT TYPES AND OBJECT-RELATIONAL TABLES

A **relational table** for "customer" can be created with the following statement:

```
CREATE TABLE INDIVIDUALS (
NAME VARCHAR2(30),
PHONE VARCHAR2(20) );
```

An object-relational table for the same data is created in two steps. First, an **object type** is defined. We call this type "O_PERSON" to distinguish it from the relational table "individuals". (The slash "/" is needed only in SQLPLUS to indicate the end of the type declaration.)

```
CREATE OR REPLACE TYPE O_PERSON AS OBJECT (
 NAME VARCHAR2(30),
 PHONE VARCHAR2(20) );
```

In a second step, an **object table** is created, which will hold the actual data (or objects).

```
CREATE TABLE PERSONS_TABLE OF O_PERSON;
```

Exercises

1. Execute these three statements in SQLPLUS.

2. Oracle provides data dictionary to examine the definitions. USER Look at the and at the description of the tables and the object type ("DESCRIBE ..."). Try these out. Write a note on the result of each statement.

```
DESC USER_TYPE_ATTRS;

SELECT *
FROM USER_TYPE_ATTRS
WHERE TYPE_NAME= 'O_PERSON';
```

```
SELECT *
FROM USER_TYPES
WHERE TYPE_NAME= 'O_PERSON';

DESC PERSONS_TABLE;

SELECT *
FROM USER_OBJECT_TABLES
WHERE Table_NAME= 'PERSONS_TABLE';
```

THERE ARE OTHERS; Try these out. What do they show?

USER_INDEXES	
USER_OBJECTS	
USER_TABLES	
USER_VIEWS	
USER_TAB_COLUMNS	
USER_TAB_COLS	

2. INSERTING VALUES

Object tables can be used both in a relational manner but also in an object-relational manner. Inserting values into an object table in a relational manner (two values are inserted):

```
INSERT INTO PERSONS_TABLE VALUES (
 'JOHN SMITH',
'1-800-555-1212' );
```

Inserting values in an object-relational manner (one value is inserted, but this one value is an object of type "person", which has itself two values):

```
INSERT INTO PERSONS_TABLE VALUES (
O_PERSON ('MARY SMITH',
          '1-800-555-1212')
);
```

Note: If types are nested, i.e., one type is used to create another type, then the object-relational insertion must be used for the nested types!

Exercises:

3. Insert five rows into "individuals" and into "persons_table". For the object table "person_table" try both methods of insertion.

4. Create a type "o_job" with four columns: "jobtitle" of datatype VARCHAR(20) and "job_ID", "salary_amount" and "years_of_experience" of datataype INTEGER. Create an object table "job_table" for this type. Insert 5 rows into this table using the object relational insertion method. Check the definitions of the object type and and object table in the data dictionary

3. OBJECT TYPES AS USER-DEFINED DATATYPES

It was described how object tables can be created that correspond to object types. In these tables each row represents one object (**row objects**). It is also possible to use object types as user-defined datatypes similar to how the predefined Oracle data-types are used. This means that objects can occupy table columns or can serve as attributes for other objects. These are called **column objects** and are described in this section.

For example, in a relational table, address information could look like this:
(streetname, Snumber, city, postal_code)

But this does not express the fact that street and number are more **closely related** than street and city. In an object table, the same information can be stored as

((streetname, Snumber), city, postal_code)

The following code shows how this is done. Note that this is the same kind of CREATE TYPE definition as used for "person". But this time there is no "street_table" created. Instead "street" is used as a datatype in "address" . In this case we have nested our types.

```
CREATE OR REPLACE TYPE O_STREET AS OBJECT (
SNAME                VARCHAR2(30),
SNUMBER              NUMBER );
```

```
CREATE OR REPLACE TYPE O_ADDRESS AS OBJECT (
STREET_AND_NUMBER    O_STREET,
CITY                 VARCHAR2(30),
POSTAL_CODE          VARCHAR2(8)
);
```

You can use your object types as user defined types in columns in a relational table. Note how the column home_address is using the user defined types address in the staff table.
In other words, rather that creating an object table like we saw earlier, we ceated a relational table where one of the columns is an object type. On object relational parlance, HOME_ADDRESS holds a column object.

```
CREATE TABLE STAFF(
ID                   INTEGER          PRIMARY KEY,
NAME                 VARCHAR2(30)     NOT NULL,
HOME_ADDRESS         O_ADDRESS        NOT NULL
);
```

Exercises

5. Create the above object types and table.
6. Insert 2 rows into your relational table. Do not forget to use your constructors. Watch your nestings!

4. DROPPING TYPES AND TABLES

Object types and object tables can be dropped and deleted in the usual manner ("DROP TABLE ...", "DROP TYPE ...") but a type or table cannot be dropped while some other type or table depends on it.

Exercises:

7. Drop types “O_PERSON” and “O_ADDRESS” and “staff” table. Also drop "O_PERSON" and "person_table".

8. Create an object type "O_ADDRESS" that contains sname, snumber, flat number, city, postal code, province and country in a manner so that *street name, street number and flat number* are “**closely related”** (i.e. has it’s own object type).

9. Create a type "O_PERSON" which contains first name, middle initial, last name, phone (business, home, mobile) and address (from previous exercise). **Make sure that first name, middle initial, last name are closely related and that the phone numbers are closely related to each other.**

10. Create an object table "student_table" that corresponds to "O_PERSON".

11. Insert five rows of data into the student table (Note: you'll have to use the object-relational form of insertion.)