**OBJECT RELATIONAL LAB (1)**

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

**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 "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 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 PERSON;

**Exercises**

1. Execute these three statements in SQLPLUS.

2. Look at the catalog ("SELECT \* FROM cat") and at the description of the tables and the object type ("DESCRIBE ...").

**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 (  
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:**

1. Insert five rows into "individuals" and into "persons\_table". For the object table "person\_table" try both methods of insertion.

INSERT INTO INDIVIDUALS VALUES (‘JOHN’,’087-9767543’); AND SO ON…

INSERT INTO PERSONS\_TABLE VALUES ('PETER','085-2234311');

INSERT INTO PERSONS\_TABLE VALUES ('PAUL','087-5637818');

INSERT INTO PERSONS\_TABLE VALUES ( PERSON('LUKE','085-90934311'));

INSERT INTO PERSONS\_TABLE VALUES ( PERSON('MATT','085-78834311'));

INSERT INTO PERSONS\_TABLE VALUES ( PERSON('FRED','085-90967651'));

COMMIT:

1. Create a type "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.

CREATE TYPE JOB AS OBJECT (

JOBTITLE VARCHAR(20),

JOB\_ID INTEGER,

SALARY\_AMOUNT INTEGER,

YEARS\_OF\_EXPERIENCE INTEGER );

/

CREATE TABLE JOB\_TABLE OF JOB;

INSERT INTO JOB\_TABLE VALUES (JOB('ENGINEER', 0, 30000,4));

INSERT INTO JOB\_TABLE VALUES ( JOB('PROGRAMMER', 1, 35000,3));

INSERT INTO JOB\_TABLE VALUES (JOB('DATA ANALYST', 2, 20000,15));

INSERT INTO JOB\_TABLE VALUES (JOB('DESIGNER', 3, 25000,2));

INSERT INTO JOB\_TABLE VALUES ( JOB('ENGINEER', 4, 33000,5));

COMMIT;

**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. That 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 STREET AS OBJECT (  
SNAME VARCHAR2(30),  
SNUMBER NUMBER );  
 /

CREATE OR REPLACE TYPE ADDRESS AS OBJECT (

STREET\_AND\_NUMBER **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.

CREATE TABLE STAFF(

ID INTEGER PRIMARY KEY,

NAME VARCHAR2(30) NOT NULL,

HOME\_ADDRESS **ADDRESS**  NOT NULL

);

**Exercises**

1. Create the above object types and table.
2. Insert 2 rows into your relational table. Watch your nestings!

INSERT INTO STAFF VALUES(1,'JAMES JONES', ADDRESS(STREET('BELGARD RD', 1), 'DUBLIN','24'));

INSERT INTO STAFF VALUES(2,'PETER HAYES', ADDRESS(STREET('PARNELL STREET', 101), 'DUBLIN','1'));

COMMIT;

**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. Anything that depends on table X must be dropped before table X can be dropped.

**Exercises:**

7. Drop types “street” and “address” and “staff” table. Also drop "person" and "person\_table".

DROP TABLE PERSONS\_TABLE;

DROP TABLE STAFF;

DROP TYPE ADDRESS;

DROP TYPE STREET;

DROP TYPE PERSON;

8. Create an object type "address" that contains sname, snumber, flat number, city, postal code, province and country in a manner so that street name, number and flat number are “closely related” (i.e. has it’s own object type). Province and country are also “closely related” i.e. you need separate object types for Street details and Geo details to start with and of course an object type address.

CREATE OR REPLACE TYPE STREET AS OBJECT (

SNAME VARCHAR2(30),

SNUMBER NUMBER,

FLATNUMBER VARCHAR2(5));

/

CREATE or replace TYPE ADDRESS AS OBJECT (

STREET\_AND\_NUMBER STREET,

CITY VARCHAR2(30),

POSTAL\_CODE VARCHAR2(8),

PROVINCE VARCHAR2(30),

COUNTRY VARCHAR2(30));

/

9. Create a type "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.

CREATE OR REPLACE TYPE PHONE AS OBJECT (

HOMEPHONE VARCHAR(15),

BUSINESSPHONE VARCHAR2(15),

MOBILEPHONE VARCHAR2(15));

/

CREATE OR REPLACE TYPE NAME AS OBJECT (

FIRSTNAME VARCHAR2(15),

MIDDLE\_INITIAL VARCHAR2(6),

LASTNAME VARCHAR2(15));

/

CREATE OR REPLACE TYPE PERSON AS OBJECT (

PNAME NAME,

PPHONE PHONE,

PADDRESS ADDRESS);

/

10. Create an object table "person\_table" that corresponds to "person".

CREATE TABLE PERSON\_TABLE OF PERSON;

10. Insert five rows of data into "person". (Note: you'll have to use the object-relational form of insertion.) Here’s 3.

INSERT INTO PERSON\_TABLE VALUES (

PERSON( NAME('JOHN', 'R', 'SMITH'),

PHONE('123-4567', NULL,'73746-56'),

ADDRESS(STREET('MARY ST', 3, '11A'),

'DUBLIN', '1', 'LEINSTER','IRELAND')));

/

INSERT INTO PERSON\_TABLE VALUES (

PERSON( NAME('MARY', NULL, 'MILLER'),

PHONE('354-5643', '453-5746','73346-56'),

ADDRESS(STREET('GRAFTON ST.',212, NULL),

'DUBLIN', '2', 'LEINSTER','IRELAND')));

/

INSERT INTO PERSON\_TABLE VALUES (

PERSON( NAME('MARY', 'S', 'MILLER'),

PHONE('322-8484', NULL,'645-2929'),

ADDRESS(STREET('OXFORD STREET',443, NULL),

'LONDON', 'W10', 'LONDON','UK')));

/

COMMIT: