---

## 1.1 Creating object types and object-relational tables

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

```
CREATE TABLE people (
    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 object-relational table "people". (The slash "/" is needed only in SQLPLUS to indicate the end of the type declaration.)

```
CREATE 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 person_table OF person;
```

**Exercises:**

It may be a good idea if you save some of the SQL statements that you develop in the exercises in a text file on your workstation (by copying into a Wordpad or Notepad file). Tables, such as person_table, will be used in several exercises. You may be asked to make changes to these tables later.

- 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 ...").

---

## 1.2 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 person_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 person_table VALUES (
    person ('Mary Smith',
    '1-800-555-1212')
);
```

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 "people" and into "person_table". For the object table "person_table" try both methods of insertion.
- 4. Create a type "job" with four columns: "jobtitle" of datatype VARCHAR(20) and "job_ID", "salary_amount" and "years_of_experience" of datataype INT. Create an object table "job_table" for this type. Insert 5 rows into this table.

---

## 1.3 Select statements

There are also two methods for selecting from an object table.

In a relational selection, two columns are selected. The use of aliases ("p" in this case) is not required but possible.

```
SELECT p.name, p.phone FROM person_table p
    WHERE p.name = 'John Smith';
```

In an object-relational selection, one column is selected. The VALUE() function retrieves objects and their values. In this case the use of an alias ("p") is required.

```
SELECT VALUE(p) FROM person_table p
    WHERE p.name = 'John Smith';
```

**Exercises:**

- 5. Try different select statements on both tables, such as "SELECT * FROM ...", "SELECT name FROM ...", "SELECT p.name FROM ...", "SELECT VALUE(p) FROM ...".

---

## 1.4 Object types as user-defined datatypes

In a previous section, 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:
(street, number, 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
((street, number), city, postale_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".

```
CREATE TYPE street AS OBJECT (
    sname VARCHAR2(30),
```

```
    snumber NUMBER );
/

CREATE TYPE address AS OBJECT (
    street_and_number street,
    city VARCHAR2(30),
    postal_code VARCHAR2(8));
/
```

**Exercises:**

- 6. Create an object type "address" that contains street name, number, flat number, city, postal code, and country in a manner so that street name, number and flat number are closely related and country is separate from all the other attributes.

---

## 1.5 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 "person" and "person_table".
- 8. Create a type "person" which contains first name, middle initial, last name, phone (business, home, mobile) and address (from exercise 6). Make sure that first name, middle initial, last name are closely related and that the phone numbers are closely related to each other.
- 9. Create an object table "person_table" that corresponds to "person".
- 10. Insert five rows of data into "person". (Note: you'll have to use the object-relational form of insertion.)

---