4.1 About attribute and type names

There seems to be a difference between tables and types with respect to attribute names. It is not possible to create types where an attribute name is the same as the name of its type. For example,

```
CREATE TYPE somename (person person);
```

causes an error message. But

```
CREATE TABLE somename (person person);
```

is ok. The same problem holds for references (person REF person).

To be on the safe side, it is probably always a good idea to choose distinctive, meaningful names for attributes and types and to use different names for types, tables and attributes.

4.2 Collection Types

Oracle supports two collection datatypes: varrays and nested tables. Varrays are variable-length ordered lists (or arrays). A maximum size of the varray must be specified but can be changed later. Nested tables are tables within tables. In contrast to varrays, nested tables are unordered lists. Varrays and nested tables are defined differently but very similar to use.

For example, people can have several phone numbers. Different categories of phone numbers, such as mobile phone versus home phone, should be stored in different columns. But different numbers of the same category (e.g., if someone has two mobile phones) could be stored in a single column in a varray or nested table.

Using varrays, a list of phone numbers can be stored in a column "phone" of a table company1. Assuming that no company would have more than 10 numbers, the varray is limited to 10 elements (VARRAY(10)).

```
CREATE TYPE phone_array AS VARRAY(10) OF VARCHAR2(12);
CREATE TABLE company1 (
    name VARCHAR2(20),
    phone phone_array
);
INSERT INTO company1 VALUES (
    'abc', phone_array('243-4758','485-2534')
);
```

name	phone
'abc'	'243-4758' '485-2534'

The following code shows the same example but uses a nested table. In the nested table, there is no limit of 10 numbers. But it has to be declared that the column "phone" of "company2" is a nested table. A storage table phone_nr_table, which stores the actual values, must be named (although this table cannot be used for anything). In this case, phone_nested is a table of just one column (VARCHAR). But types with multiple columns can also be nested (see Exercise 25).

```
CREATE TYPE phone_nested AS TABLE OF VARCHAR2(12);
CREATE TABLE company2 (
    name VARCHAR2(20),
    phone phone_nested
    )
    NESTED TABLE phone STORE AS phone_nr_table;
INSERT into company2 VALUES (
    'abc', phone_nested('243-4758','485-2534')
);
```

Exercises

- 25. Create a type phone_code that has area_code and local_number as attributes. Create a type phone_numbers as a table of phone_code. Create a table company3 that has the attributes name and phone, where phone is of type phone_numbers.

name	phone	
	area_code	**localnumber**
'abc'	'0131'	'243-4758'
	'0131'	'485-2534'

- 26. Insert several rows into company3. Note that in contrast to the insert statement for company2, you now have one more level "phone_code" which must be used when constructing the values.
- 27. Some questions to think about: Which Normalform is violated when using varrays or nested tables? An Oracle consulting website discusses "planned data denormalization". It claims that the only purpose of normalization is saving disk space. What are other purposes that are sometimes attributed to normalization? Do they still pose a problem?

4.3 Multi-level collection types and other advanced features

- Nesting and varrays can be applied repeatedly and form so-called "multilevel collection types". That means that varrays of nested tables, nested tables of nested tables, varrays of user types that may have collection types as attributes and so on can all be created. (This will not be discussed further in this tutorial.)
- It is possible to use references within nested tables. But in that case insertion of rows requires some advanced functions, which have not yet been discussed in this tutorial. A simple trick is not to use references within nested tables but instead to use standard relational techniques (i.e., foreign keys).
- The NESTED TABLE statement is only used for tables not for types because it refers to the storage requirements for the data. Thus if a type has an attribute which is a nested table, the NESTED TABLE statement is not mentioned in the creation of the type but instead in the creation of its object table.

Optional Exercise:

- Why is it difficult to insert rows into a nested table which has references?

4.4 Select statements for nested tables

There are two types of queries: queries that retrieve objects in a nested format showing all their types and queries that show the data in an un-nested format (just the data, not the types).

The following examples work for both company1 (using varrays) and company2 (using nested tables). To select all data and types:

```
SELECT * from company1;
```

The SQL function TABLE() un-nests the nested table. In the following example, "c" is an alias for company1, "c.phone" is the nested table in company1. The un-nested c.phone table is assigned its own alias "t".

```
SELECT c.name, t.* from company1 c, TABLE(c.phone) t;
```

An equivalent version is

```
SELECT name, t.* from company1, TABLE(SELECT phone FROM company1) t;
```

Oracle provides a default column name, COLUMN_VALUE, for the column of a nested table that does not have a name:

```
SELECT t.COLUMN_VALUE from company1 c, TABLE (c.phone) t;
```

Exercises

- 28. Use the table company3 from the previous exercises. Write SQL statements for the following: Select all phone numbers which belong to a given company name. Select all companies which have a certain area_code. Select all localnumbers.