5.1 Ordinary Relations and Object Relations

To convert ordinary relational tables into object tables one can either create new tables and drop the old ones or one can create object-relational views for the old tables but keep the data in the old tables.

As an example, consider a relational table:

```
CREATE TABLE people (
    ID number PRIMARY KEY,
    name VARCHAR2(30),
    phone VARCHAR2(20) );
INSERT into people VALUES (0, 'Smith', '546-4364');
INSERT into people VALUES (1, 'Miller', '556-4374');
INSERT into people VALUES (2, 'Jones', '536-4386');
```

A corresponding object type is "people_type" below. The type "phone_nested" from last week's exercises is used so that a person can have more than one phone number.

```
CREATE TYPE phone_nested AS TABLE OF VARCHAR2(12);
CREATE TYPE people_type AS OBJECT (
    person_ID number,
    name VARCHAR2(30),
    phone_list phone_nested);
```

The first possibility to define an object-relational table is to insert the data from "people" into "people_object_table". When inserting the data into the object-relational table the types need to be specified, such as "phone_nested(phone)".

```
CREATE TABLE people_object_table OF people_type
    NESTED TABLE phone_list STORE AS p_table;
INSERT into people_object_table
    SELECT ID, name, phone_nested(phone) from people;
```

The second possibility to define an object-relational table is to create an object view. Because an object view does not duplicate the data, it is not necessary to define a storage table (such as p_table). But a column must be specified that is used as an object identifier. This is normally the column that corresponds to the primary key of the relational table.

```
CREATE VIEW people_object_view OF people_type
    WITH OBJECT IDENTIFIER (person_ID) AS
    SELECT ID, name, phone_nested(phone) AS phone_list from people;
```

Exercises:

- 29. Create a relational table "department", which has columns dno, dname, dstreet, dstreetnumber, dcity and dpostalcode. Create a type for the address information (or use the address type from previous exercises). Create a type "dept_type" with columns deptno, deptname and deptaddress. Create an object view dept_view that views the data from the relational table.

- 30. Insert three rows into the department table. Use the object view to view the rows. Can you use the object view to insert further rows?

5.2 Using Nested Tables in Views

The previous example showed how to define a view with a nested table column (phone_list) but not how to insert multiple phone numbers into that column. To insert multiple phone numbers for a person, the phone numbers must be stored in a separate table, such as phone_nrs below.

```
CREATE TABLE phone_nrs (
    ID number,
    phone VARCHAR2(20) );
INSERT into phone_nrs VALUES (0, '546-4364');
INSERT into phone_nrs VALUES (0, '546-4123');
INSERT into phone_nrs VALUES (1, '556-4374');
INSERT into phone_nrs VALUES (2, '536-4386');
```

The view people_object_view2 combines data from the relational table people and from the relational table phone_nrs. Because multiple rows from phone_nrs can belong to one row from people, the functions CAST(MULTISET()) must be used as shown in the example.

```
CREATE VIEW people_object_view2 OF people_type
    WITH OBJECT IDENTIFIER (person_ID) AS
    SELECT p.ID, p.name,
    CAST(MULTISET (SELECT phone FROM phone_nrs n
    WHERE n.ID = p.ID) AS phone_nested)
    FROM people p;
```

Exercise:

- 31. Create a relational table employees with columns empID, empname and deptno. Insert a few rows into this table. Create an object type employee_t with columns eID and ename. Create a type employee_list_t which is a nested table of employee_t. Create a type dept_t with columns deptno, deptname and emp_list where emp_list is of type employee_list_t. Finally, create an object view that combines the department data from the previous exercises with the employees from the employees table.

5.3 Further Information about Object Views

There is much more to know about object views. For example, in some circumstances it is not possible to use object views to update data. Instead triggers must be used.

References to rows in object views can be created with MAKE_REF. This allows for creating inverse relationships or circular references.

Object views can form a hierarchy (using UNDER).

Details about such advanced features can be found in Applying an Object Model to Relational Data. But these advanced features will not be covered in this tutorial.