Object-relational modeling in Oracle 9i

Unfortunately, the syntax (and to some degree also the expressivity) of Oracle’s implementation of object-relational features differs from the general description given in the lecture. The following example scenario should give you enough hints to serve as a guideline to solve the last task of assignment 8.

Scenario
We want to model the following scenario: We consider Employees (with names) who can be (partially) subdivided into Workers (with a start date of the contract) and Managers (with a salary attribute). Workers work on (arbitrarily many) Projects (with a name), a Project involves arbitrarily many Workers. Furthermore, a Project is managed by exactly one Manager. A Manager (who can be reached by many telephone numbers) can manage many Projects.

This scenario is illustrated by the following ER diagram.

Object-Relational-Modelling
First, we create object types for each of the given entity types.

```
CREATE TYPE employee_t AS OBJECT (name varchar2(100)) NOT FINAL;
Types, that can be specialized, must be declared as NOT FINAL.

CREATE TYPE worker_t UNDER employee_t (contractDate date);

CREATE TYPE telephone_t AS TABLE OF varchar2(30);

CREATE TYPE manager_t UNDER employee_t
    (salary integer, telephone telephone_t);

CREATE TYPE workers_t AS TABLE OF REF worker_t;
```
CREATE TYPE project_t AS OBJECT {
    name varchar2(100),
    managedBy REF manager_t,
    workers workers_t
};

As you can see, we decided to model the relationship “manages” by the managedBy reference in the Project_t class and implemented the worksOn relationship by listing a collection (realized as a nested table) of workers in a Project_t object. Please notice that – as Oracle does not support automatically updated “inverse-Relationships” – we decided to realize links between objects as listings of link partners on one side only.

Now we create tables to store objects of the above defined types.
CREATE TABLE worker OF worker_t;
CREATE TABLE manager OF manager_t
    NESTED TABLE telephone STORE AS telephone_nt;
In Oracle you have to explicitly give the name of the nested table (telephone_nt) that stores data of collection types.

CREATE TABLE project OF project_t
    NESTED TABLE workers STORE AS workers_nt;
ALTER TABLE project ADD (SCOPE FOR (managedBy) IS manager);
ALTER TABLE workers_nt ADD (SCOPE FOR (COLUMN_VALUE) IS worker);
In addition to the table definition, the valid reference scope is constrained. Since workers_t is just a table of worker_t object the “placeholder” attribute name COLUMN_VALUE must be used.

Please keep in mind that Oracle does not guarantee the referential integrity between object references and object instances. Therefore, you might end up with so called dangling references if you e.g. delete a manager that is used in a project object.

Inserting data
Two employees and one Manager:
INSERT INTO worker VALUES ('John Brown', cast ('19-02-2002' as date));
INSERT INTO worker VALUES (worker_t('Jim Smith', cast ('11-02-2002' as date)));
INSERT INTO manager VALUES (manager_t('Big Boss', 60000, telephone_t('324-32', '321-32')));

One project that – for the first – nobody works on, but that is managed by Big Boss:
INSERT INTO project VALUES ('The big nothing',
    (SELECT REF(m) FROM manager m WHERE m.name='Big Boss'),
    Workers_t());

Assign the two defined workers to this project:
INSERT INTO TABLE( SELECT workers FROM project WHERE name='The big nothing')
    (SELECT REF(w) FROM worker w
        WHERE w.name='John Brown' OR w.name='Jim Smith');

Another project managed by Big Boss that only John Brown works on
INSERT INTO project VALUES (project_t('Small project',
    (SELECT REF(m) FROM manager m WHERE m.name='Big Boss'),
    (SELECT REF(w) FROM worker w
        WHERE w.name='John Brown' OR w.name='Jim Smith');
Some example queries

SELECT * FROM worker
Displays all attributes of the worker_t type of all objects in the worker table. By this command, the resulting table looks like a normal relation.

SELECT value(w) FROM worker w
Displays the same, but shows the objects in the form worker_t(name, date) (constructor format)

SELECT p.name, DEREF(p.managedBy) FROM project p
Shows all project names and their managers in constructor format.

SELECT p.name, DEREF(p.managedBy).name FROM project p
Show all project names together with the name of their manager.

SELECT p.name, DEREF(w.COLUMN_VALUE) FROM project p, TABLE(p.workers) w
Unnests the associated workers for each project and shows its name together with the worker’s object in constructor format (so the TABLE operator of Oracle works like the UNNEST operator presented in the lecture).

Some useful hints for your practical work with the Oracle DBMS

SELECT * FROM USER_TYPES
Lists all user defined data types

SELECT * FROM USER_ALL_TABLES WHERE TABLE_TYPE IS NOT NULL
Shows all object tables.

DROP TYPE employee
Removes user defined data type employee from the user’s schema.

The message “Warnung: Typ wurde mit Kompilierungsfehlern erstellt.” indicates an error during the type creation. Drop the type and correct the mistake!