Advanced Management of Data

Exercise 2 Topic 3:

Object-Relational Database Systems

Object Methods?

- Task: having the date of birth is nice, but often you need the age, so add a method that returns the age in years as `INTEGER`

```
CREATE OR REPLACE FUNCTION age(person persons) RETURNS INTEGER AS $$
BEGIN
  RETURN EXTRACT(YEAR from AGE(person.dateofbirth));
END;
$$ LANGUAGE plpgsql;
```

- unfortunately, PostgreSQL doesn't support methods, but you can call functions with one parameter like they were attributes, so instead of

```
functionname(objectname);
```

- you can write

```
(objectname).functionname;
```

- Task: write a function that uses this syntax to return the age of a person, that is given by its name as parameter

Functions

```
CREATE OR REPLACE FUNCTION getAge(name nametype) RETURNS INTEGER AS $$
DECLARE
  person persons;
BEGIN
  SELECT * INTO person FROM persons WHERE persons.name = getAge.name;
  RETURN (person).age;
END;
$$ LANGUAGE plpgsql;
```

- now you can test your function, like

```
SELECT (('Max', 'Mustermann')).getAge;
```

- <u>Task</u>: the Doe’s got a new job abroad and therefore are moving, too, so change their address to 1 Rue Vincent d’Indy, 59650 Villeneuve-d’Ascq, France and their telephone number to +33 3 33 33 33 33

Inheritance

```
UPDATE persons
  SET address = (('Rue Vincent d''Indy', '1'),
                 ('Villeneuve-d''Ascq', '59650'), 'France'),
      telephone = '+33 3 33 33 33 33'
  WHERE (name).surname = 'Doe';
```

- they lecture at Université de Lille
  - Jane is associate professor and earns 4000 EUR per month
  - John is assistant professor and earns 3000 EUR per month
- Task: create a new table for `professors` that inherits everything from `persons` and in addition stores their `university`, `rank` and `salary`
- finally, add the Doe’s information to the new table

Inheritance
Moved

```
CREATE TABLE professors (university VARCHAR, rank VARCHAR, salary MONEY) INHERITS (persons);
```

- adding information to existing persons is a bit tricky, but can be done by moving the record from `persons` to `professors` while adding the new information

```
WITH moved AS (DELETE FROM persons WHERE name = ('Jane', 'Doe')::nametype RETURNING *)
  INSERT INTO professors
    SELECT name, address, email, telephone, dateofbirth, 'Université de Lille', 'associate professor', 4000
    FROM moved;

WITH moved AS (DELETE FROM persons WHERE name = ('John', 'Doe')::nametype RETURNING *)
  INSERT INTO professors
    SELECT name, address, email, telephone, dateofbirth, 'Université de Lille', 'assistant professor', 3000
    FROM moved;
```

- John was killed in an accident at university (now, we have a dead John Doe) and Jane quit her job as she can’t stand to work there any longer
- Task: remove John from `persons` and Jane from `professors`

Inheritance
Removed

```
DELETE FROM persons WHERE name = ('John', 'Doe')::nametype;
```

- as you can see, John is also removed from `professors`
- the same could be achieved by

```
DELETE FROM professors WHERE name = ('John', 'Doe')::nametype;
```

- so I assume most of you got Jane killed, too
- to keep Jane in `persons` and remove her ONLY from `professors` you have to write something like

```
DELETE FROM ONLY professors WHERE name = ('Jane', 'Doe')::nametype;
```

- by using `ONLY` in queries on inherited tables, just the selected table is processed and all related tables are ignored
- our `nametype` just supports forename and surname, but most people also have one or more nicknames
- Task: find a way to store several nicknames along with the other names inherited from `nametype`

Inheritance
Typed

- PostgreSQL doesn't support type inheritance but only table inheritance
- therefore you have to create a table for the type first

```
CREATE TABLE nametable OF nametype;
```

- as we won't use real inheritance here, we can just copy the structure from nametable along with the new nicknames

```
CREATE TABLE names (LIKE nametable, nickname VARCHAR[]);
```

- Task: add the Mustermanns to the new table `names`
  - Max Mustermann is nicknamed Maxl and Maxi
  - Erika Mustermann is nicknamed Rikki, Ri and Rika

Arrays

```
INSERT INTO names VALUES
  ('Max', 'Mustermann', ARRAY['Maxl', 'Maxi']),        -- use can use the ARRAY constructor syntax
  ('Erika', 'Mustermann', '{"Rikki", "Ri", "Rika"}'); -- or a literal constant to declare array values
```

- arrays can be used with every built-in or user-defined type, but domains are not yet supported
- one-dimensional arrays can also be defined with standard `ARRAY`-Syntax and also the size can be specified, but is ignored by the current implementation (so no size restrictions are enforced)

```
nickname VARCHAR ARRAY[2]
```

- it is possible to define multi-dimensional arrays, but only with the square bracket syntax

```
used BOOLEAN[][]
```