## НАУЧНО-ИССЛЕДОВАТЕЛЬСКАЯ КОРПОРАЦИЯ ИТМО



Факультет программной инженерии и компьютерной техники

# Информационные системы и базы данных Лабораторная работа № 1

Выполнил студент

Нестеров Иван Алексеевич

Группа Р33302

Преподаватель: Гаврилов Антон Валерьевич

#### Вариант 312621

Please, enter your variant: 312621

Описание предметной области, по которой должна быть построена доменная модель:

Тим внимательно разглядывал обочину дороги. Дождь хлестал со страшной силой, капли стучали по листьям, и те тряслись. Все вокруг пришло в движение. Все казалось живым. Тим осматривал буквально каждый листок... И вдруг замер! Под деревьями кто-то стоял. Тим поднял глаза и посмотрел повыше. За деревьями, по ту сторону забора, он увидел что-то огромное, шероховатое, словно кора дерева. Однако это было не дерево... Тим поправил очки и посмотрел еще выше...

Тим внимательно разглядывал обочину дороги. Дождь хлестал со страшной силой, капли стучали по листьям, и те тряслись. Все вокруг пришло в движение. Все казалось живым. Тим осматривал буквально каждый листок... И вдруг замер! Под деревьями кто-то стоял. Тим поднял глаза и посмотрел повыше. За деревьями, по ту сторону забора, он увидел что-то огромное, шероховатое, словно кора дерева. Однако это было не дерево... Тим поправил очки и посмотрел еще выше...

#### Задание:

- 1. На основе предложенной предметной области (текста) составить ее описание. Из полученного описания выделить сущности, их атрибуты и связи.
- 2. Составить инфологическую модель.
- 3. Составить даталогическую модель. При описании типов данных для атрибутов должны использоваться типы из СУБД PostgreSQL.
- 4. Реализовать даталогическую модель в PostgreSQL. При описании и реализации даталогической модели должны учитываться ограничения целостности, которые характерны для полученной предметной области.
- 5. Заполнить созданные таблицы тестовыми данными.

### Ход работы:

Описание предметной области. Три сущности: дождь, листья, человек. По логике текста, сущности связаны между собой следующим образом: человек - one-to-many - листья - many-to-one - дождь, так как человек смотрит на листья, которые испытывают на себе воздействие дождя (трясутся). У каждой из сущностей есть набор атрибутов. Опишем их все и придумаем ограничение целостности на атрибуты.

### Ограничения целостности:

Вода: температура воды может быть (0; 100) градусов

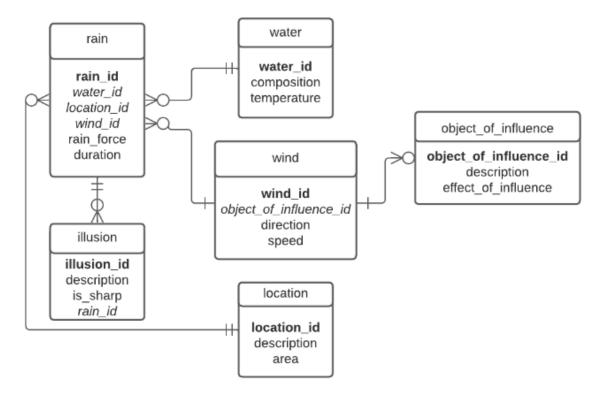
Локация: площадь (0; 20) квадратных километров.

Ветер: скорость - положительное число (0; 120] м/с

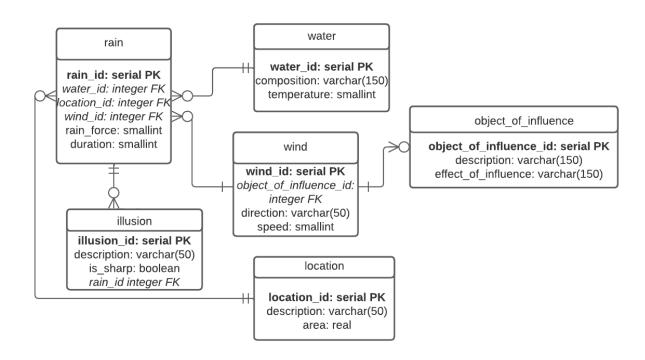
Дождь: сила [0;5] баллов, длительность (0; 365] дней.

Инфологическая (семантическая) модель: ER-диаграмма. Изобразим ER-диаграмму на семантическом уровне приближения - без детализации о типах данных, например, лишь

сущности, связи между ними, и атрибуты:



Теперь создадим даталогическую модель под PostgreSQL:



Классификация сущностей:

Стержневая: дождь, вода, локация, объект воздействия;

Ассоциативная: ветер, отражает связь дождя и объекта воздействия;

Характеристическая: иллюзия - дополнительно характеризует дождь.

Создадим вышеописанные сущности, опишем связи между ними и ограничения целостности при помощи написанного для этого sql-скрипта:

```
CREATE TABLE IF NOT EXISTS water (
    id serial PRIMARY KEY,
    composition varchar(150) NOT NULL,
    temperature smallint NOT NULL,
    CHECK(temperature > 0 AND temperature < 100)</pre>
);
CREATE TABLE IF NOT EXISTS object of influence (
    id serial PRIMARY KEY,
    description varchar(150) NOT NULL,
    effect_of_influence varchar(150) NOT NULL
);
CREATE TABLE IF NOT EXISTS location (
    id serial PRIMARY KEY,
    description varchar(50) NOT NULL,
    area real NOT NULL,
    CHECK(area > 0 AND area <= 20)</pre>
);
CREATE TABLE IF NOT EXISTS wind (
    id serial PRIMARY KEY,
    direction varchar(50) NOT NULL,
    speed smallint NOT NULL,
    object_of_influence_id integer REFERENCES object_of_influence,
    CHECK(speed > 0 and speed <= 120)
CREATE TABLE IF NOT EXISTS rain (
    id serial PRIMARY KEY,
    rain force smallint NOT NULL,
    duration smallint NOT NULL,
    water id integer UNIQUE REFERENCES water,
    location_id integer UNIQUE REFERENCES location,
    wind_id integer REFERENCES wind,
    CHECK((rain force >= 0 AND rain force <= 5)</pre>
    AND (duration > 0 AND duration <= 365))
);
CREATE TABLE IF NOT EXISTS illusion (
    id serial PRIMARY KEY,
    description varchar(50) NOT NULL,
    is_sharp boolean NOT NULL,
    rain_id integer UNIQUE REFERENCES rain
```

Отправим скрипт работать на helios. Что было в схеме до выполнения скрипта:

```
\Theta
Microsoft Windows [Version 10.0.19044.1889]
(c) Microsoft Corporation. All rights reserved.
C:\Users\Иван Heстеров>ssh -p 2222 s312621@helios.se.ifmo.ru
Last login: Tue Sep 13 17:55:13 2022 from 77.234.209.96
FreeBSD 13.0-STABLE (amd64) #0 0d0eb707b: Thu Apr 28 15:19:16 UTC 2022
Welcome to FreeBSD!
[s312621@helios ~]$ psql -h pg -d studs -W
Пароль:
psql (14.2)
.
SSL-соединение (протокол: TLSv1.3, шифр: TLS_AES_256_GCM_SHA384, бит: 256, сжатие: выкл.)
Введите "help", чтобы получить справку.
studs=> \dt;
               Список отношений
 Схема
                                     Владелец
                                       s336473
 public
          Coordinate
                            таблица |
                                       s336473
 public
          Dragon
                            таблица
 public
          Person
                            таблица
                                       s336473
 public
          Users
                            таблица |
                                       s336473
          organizations
 public
                            таблица
                                       s335159
 public
          users
                            таблица |
                                       s335159
(6 строк)
studs=>
```

#### Что стало после:

```
[s312621@helios ~/inf-systems-and-databases]$ psql -h pg -d studs -W
Пароль:
psql (14.2)
SSL-соединение (протокол: TLSv1.3, шифр: TLS AES 256 GCM SHA384, бит: 256, сжатие: выкл.)
Введите "help", чтобы получить справку.
studs=> \dt;
                 Список отношений
  Схема
                  ИМЯ
                                  Тип
                                         Владелец
 public | Coordinate
                                таблица | s336473
 public
                                таблица
                                          s336473
          Dragon
 public
                                таблица | s336473
          Person
 public
                                таблица | s336473
          Users
 public
          organizations
                                таблица | s335159
public | users
                                таблица | s335159
 s312621 | illusion
                                таблица | s312621
                                таблица | s312621
 s312621 | location
 s312621 | object_of_influence
                                таблица | s312621
 s312621 | rain
                                 таблица | s312621
                                 таблица | s312621
 s312621
          water
 s312621 | wind
                                 таблица | s312621
(12 строк)
studs=>
```

Скрипт был запущен на исполнение извне psql консоли следующей командой:

```
psql -h pg -d studs -U s312621 -W -a -f lab1.sql
```

Заполним таблицу тестовыми данными. Например, начнем с таблицы water и попытаемся вставить сущность с нарушенной целостностью:

```
studs=> SELECT * FROM water;
id | composition | temperature
(0 строк)
studs=> INSERT INTO water (composition, temperature) VALUES ('Прохладная вода', 10);
INSERT 0 1
studs=> SELECT * FROM water;
id | composition | temperature
 1 | Прохладная вода | 10
(1 строка)
studs=> INSERT INTO water (composition, temperature) VALUES ('Теплая вода', 25);
INSERT 0 1
studs=> INSERT INTO water (composition, temperature) VALUES ('Аномальная вода', -15);
ERROR: new row for relation "water" violates check constraint "water_temperature_check"
ПОДРОБНОСТИ: Failing row contains (3, Аномальная вода, -15).
studs=> SELECT * FROM water;
id | composition | temperature
 1 | Прохладная вода | 10
2 | Теплая вода | 25
(2 строки)
```

Увидим ошибку. Ограничение целостности сработало.

Заполним БД тестовыми данными, используя простые INSERT запросы, ради интереса пытаясь периодически этими запросами нарушить целостность данных и радуясь, что функция CHECK успешно работает. Выведем все таблички:

```
studs=> SELECT * FROM rain;
id | rain_force | duration | water_id | location_id | wind_id
1 2 6 2 3
(1 строка)
studs=> SELECT * FROM illusion;
id | description | is_sharp | rain_id
1 | дерево похоже на человека | f
(1 строка)
studs=> SELECT * FROM water;
id | composition | temperature
1 | Прохладная вода |
2 | Теплая вода |
                            10
                             25
(2 строки)
studs=> SELECT * FROM wind;
id | direction | speed | object_of_influence_id
1 | NORD-EAST | 10 |
2 | NORD-WEST | 7 | 3 | NORD-WEST | 7 |
                                          1
                                          1
(3 строки)
studs=> SELECT * FROM location;
id | description | area
 3 Городская застройка 15
 4 | Городская застройка | 10
```

```
studs=> SELECT * FROM location;
          description
id
                             area
      Городская застройка
                               15
      Городская застройка
                               10
2 строки)
studs=> SELECT * FROM object_of_influence;
     description | effect_of_influence
id
  1
      дерево
                    шатается
      куст
                    обломилась ветка
(2 строки)
studs=>
```

**Вывод:** в ходе работы был изучен синтаксис языка программирования SQL, понятия атрибутов, сущностей, связей и ограничения целостности, составлены инфологическая и даталогическая ER-диаграммы, а так же освежены знания по работе с psql.