

Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Лабораторная работа по базам данных №1

Вариант № 31071

Выполнил:

Студент группы Р3106

Мельник Фёдор Александрович

Проверил:

Вербовой Александр Александрович,

Преподаватель-практик ФПИиКТ

Санкт-Петербург, 2025

Оглавление

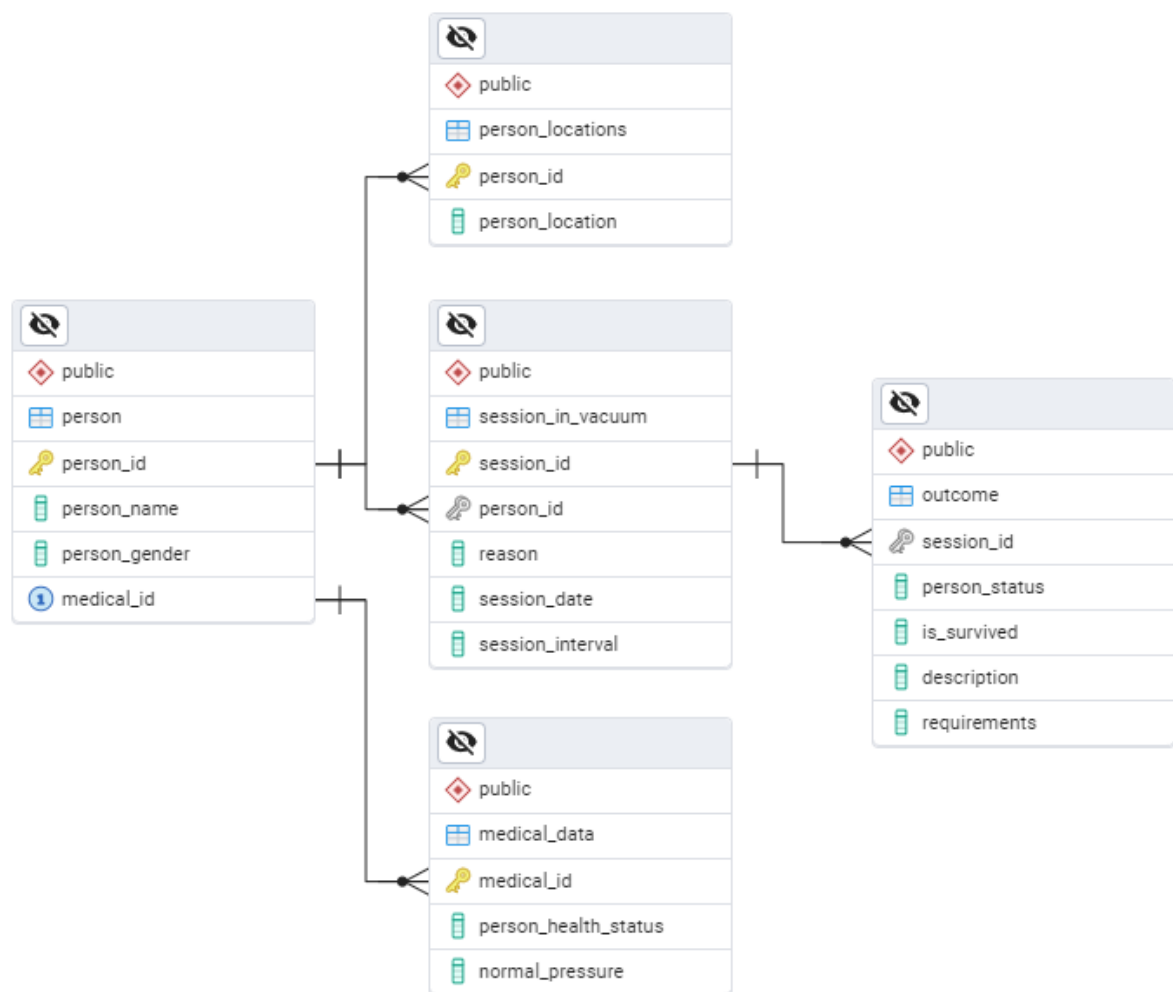
Текст задания	3
Выполнение.....	4
Функциональные зависимости.....	4
Нормальные формы.....	5
1NF	5
2NF	5
3NF	5
BCNF.....	6
Денормализация.....	9
Триггер и связанная с ним функция.....	9
Вывод.....	13

Текст задания

Для отношений, полученных при построении предметной области из лабораторной работы №1, выполните следующие действия:

- Опишите функциональные зависимости для отношений полученной схемы (минимальное множество);
- Приведите отношения в 3NF (как минимум). Постройте схему на основе 3NF (как минимум).
- Опишите изменения в функциональных зависимостях, произошедшие после преобразования в 3NF (как минимум). Постройте схему на основе 3NF;
- Преобразуйте отношения в BCNF. Докажите, что полученные отношения представлены в BCNF. Если ваша схема находится уже в BCNF, докажите это;
- Какие денормализации будут полезны для вашей схемы? Приведите подробное описание.

Придумайте триггер и связанную с ним функцию, относящиеся к вашей предметной области, согласуйте их с преподавателем и реализуйте на языке PL/pgSQL.



Выполнение

Функциональные зависимости

person: person_id \rightarrow person_name, person_id, medical_id

medical_data: medical_id \rightarrow person_health_status, normal_pressure

session_in_vacuum: session_id \rightarrow person_id, session_date, session_interval, reason

outcome: session_id \rightarrow description, is_survived, person_status, requirements

person_locations: person_id \rightarrow person_location

Нормальные формы

1NF

Первая нормальная форма выполняется, так как все значения являются атомарными

2NF

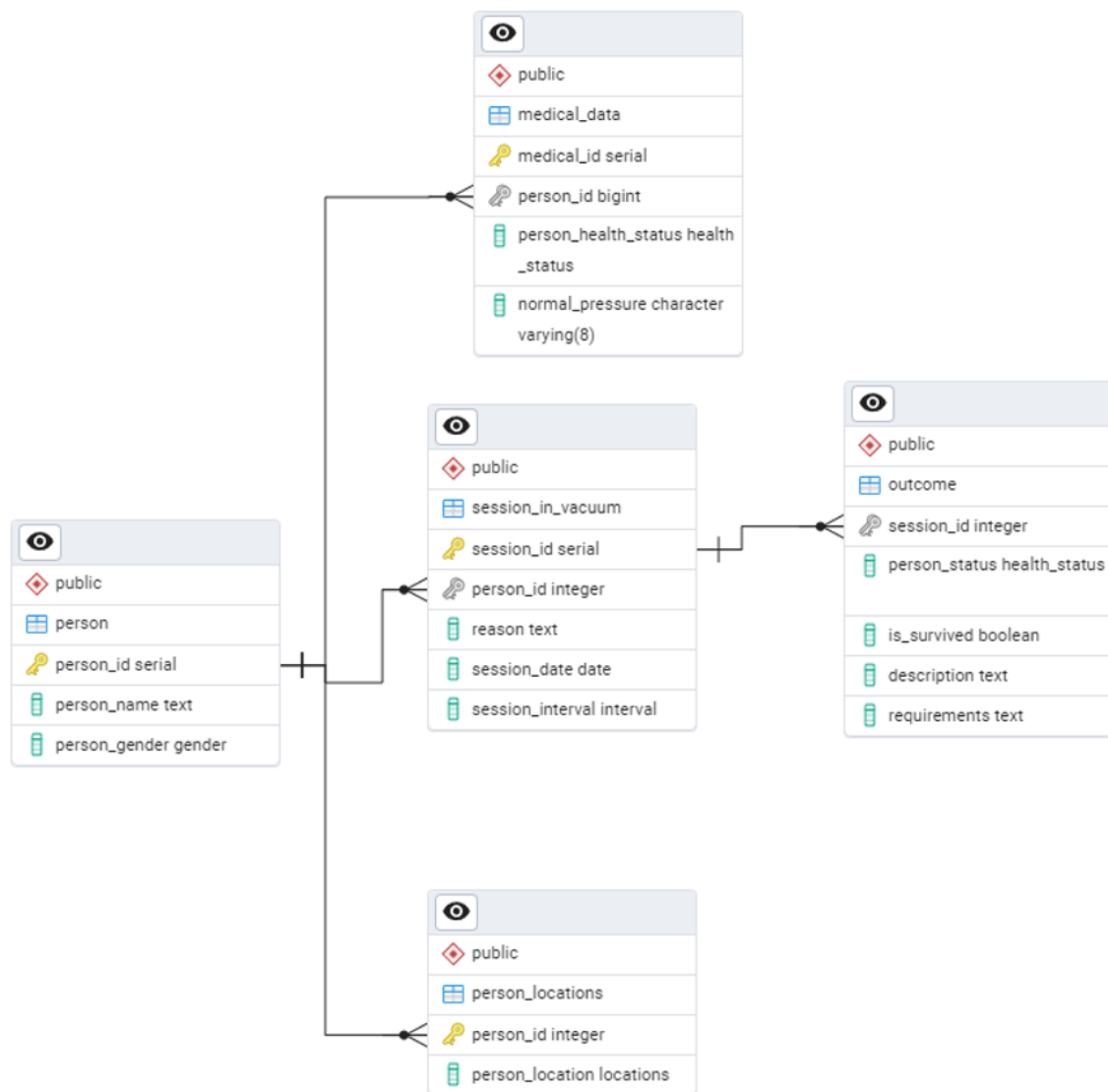
Вторая нормальная форма выполняется, так как отношения находятся в 1NF и все атрибуты зависят от всего первичного ключа, а не его части

3NF

Третья нормальная форма не выполняется, так как присутствуют транзитивные зависимости: $\text{person}(\text{person_id} \rightarrow \text{medical_id})$ и $\text{medical_data}(\text{medical_id} \rightarrow \text{person_health_status})$

Исправим, убрав зависимость $\text{person}(\text{person_id} \rightarrow \text{medical_id})$ и добавив $\text{medical_data}(\text{medical_id} \rightarrow \text{person_id})$

Новая схема:



BCNF

BCNF выполняется, так как отношения находятся в 3NF (после изменений) и для каждой функциональной зависимости $X \rightarrow Y$, X является суперклассом ключа (суперключом)

Реализация схемы после изменений

```
BEGIN;

CREATE TYPE gender AS ENUM ('мужской', 'женский');

CREATE TABLE IF NOT EXISTS person
(
    person_id SERIAL PRIMARY KEY,
    person_name TEXT NOT NULL,
    person_gender gender NOT NULL
);

CREATE TYPE health_status AS ENUM ('мёртв', 'ранен', 'травмирован', 'здоров');

CREATE TABLE IF NOT EXISTS medical_data
(
    medical_id SERIAL PRIMARY KEY,
    person_id BIGINT UNIQUE REFERENCES person(person_id) ON DELETE
    CASCADE,
    person_health_status health_status NOT NULL,
    normal_pressure VARCHAR(8) NOT NULL
);

CREATE FUNCTION validate_pressure()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.normal_pressure !~* '^[1-9][0-9]{0,3}/[1-9][0-9]{0,2}$'
    THEN
        RAISE EXCEPTION 'Неверный формат normal_pressure';
    END IF;
```

```

        RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER check_pressure BEFORE INSERT OR UPDATE ON medical_data
FOR EACH ROW EXECUTE FUNCTION validate_pressure();

CREATE TYPE locations AS ENUM ('вакуум', 'нормальное');

CREATE TABLE IF NOT EXISTS person_locations (
    person_id INT UNIQUE REFERENCES person(person_id) ON DELETE CASCADE,
    person_location locations NOT NULL,
    PRIMARY KEY (person_id)
);

CREATE TABLE IF NOT EXISTS session_in_vacuum (
    session_id SERIAL PRIMARY KEY,
    person_id INT NOT NULL REFERENCES person(person_id) ON DELETE CASCADE,
    reason TEXT,
    session_date DATE,
    session_interval INTERVAL
);

CREATE TABLE IF NOT EXISTS outcome (
    session_id INT NOT NULL UNIQUE REFERENCES session_in_vacuum(session_id)
ON DELETE CASCADE,
    person_status health_status NOT NULL,
    is_survived BOOLEAN NOT NULL,
    description TEXT,
    requirements TEXT
);

INSERT INTO person(person_name, person_gender) VALUES

```

```
    ('Алекс', 'мужской'),
    ('Майк', 'мужской'),
    ('Дрожия', 'женский');
INSERT INTO medical_data VALUES
    ('12345', '1', 'здоров', '120/79'),
    ('54321', '2', 'травмирован', '121/80'),
    ('89654', '3', 'мёртв', '135/80');
INSERT INTO person_locations VALUES
    ('1', 'вакуум'),
    ('2', 'нормальное'),
    ('3', 'нормальное');
INSERT INTO session_in_vacuum(person_id, reason, session_date, session_interval) VALUES
    ('1', 'эксперимент', '2025-02-21', '54 seconds'),
    ('2', 'несчастный случай', '1999-01-23', '4 minutes 56 seconds'),
    ('3', 'убийство', '2003-04-12', '3 hours 12 minutes 14 seconds');
INSERT INTO outcome VALUES
    ('1', 'здоров', TRUE, NULL, 'рекомпрессия'),
    ('2', 'травмирован', TRUE, 'удалось спасти, но остался частично парализованным из-за воздушной эмболии', null),
    ('3', 'мёртв', FALSE, 'летальный исход', NULL);
END;
```


Денормализация

Для моей схемы могут быть полезны следующие денормализации:

- Объединение таблиц. Данная денормализация помогает оптимизировать запросы, уменьшая количество JOIN. Например, можно объединить person и person_locations
- Дублирование данных. Данная денормализация позволяет быстрее получать данные от запросов, избегая длинные «пути» между сущностями. Например, можно продублировать person_health_status из medical_data в person

Реализация схемы после денормализаций

```
BEGIN;

CREATE TYPE gender AS ENUM ('мужской', 'женский');

CREATE TYPE locations AS ENUM ('вакуум', 'нормальное');

CREATE TYPE health_status AS ENUM ('мёртв', 'ранен', 'травмирован', 'здоров');

CREATE TABLE IF NOT EXISTS person
(
    person_id SERIAL PRIMARY KEY,
    person_name TEXT NOT NULL,
    person_gender gender NOT NULL,
    person_location locations NOT NULL,
    person_health_status health_status NOT NULL
);

CREATE TABLE IF NOT EXISTS medical_data
(
    medical_id SERIAL PRIMARY KEY,
    person_id BIGINT UNIQUE REFERENCES person(person_id) ON DELETE
    CASCADE,
    person_health_status health_status NOT NULL,
    normal_pressure VARCHAR(8) NOT NULL
);

CREATE FUNCTION validate_pressure()
```

```

RETURNS TRIGGER AS $$
BEGIN
    IF NEW.normal_pressure !~* '^[1-9][0-9]{0,3}/[1-9][0-9]{0,2}$'
    THEN
        RAISE EXCEPTION 'Неверный формат normal_pressure';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_pressure BEFORE INSERT OR UPDATE ON medical_data
FOR EACH ROW EXECUTE FUNCTION validate_pressure();

CREATE TABLE IF NOT EXISTS session_in_vacuum (
    session_id SERIAL PRIMARY KEY,
    person_id INT NOT NULL REFERENCES person(person_id) ON DELETE CASCADE,
    reason TEXT,
    session_date DATE,
    session_interval INTERVAL
);

CREATE TABLE IF NOT EXISTS outcome (
    session_id INT NOT NULL UNIQUE REFERENCES session_in_vacuum(session_id)
ON DELETE CASCADE,
    person_status health_status NOT NULL,
    is_survived BOOLEAN NOT NULL,
    description TEXT,
    requirements TEXT
);

INSERT INTO person(person_name, person_gender, person_location, person_health_status)
VALUES

```

```
      ('Алекс', 'мужской', 'вакуум', 'здоров'),  
      ('Майк', 'мужской', 'нормальное', 'травмирован'),  
      ('Дроджия', 'женский', 'нормальное', 'мёртв');  
INSERT INTO medical_data VALUES  
      ('12345', '1', 'здоров', '120/79'),  
      ('54321', '2', 'травмирован', '121/80'),  
      ('89654', '3', 'мёртв', '135/80');  
INSERT INTO session_in_vacuum(person_id, reason, session_date, session_interval) VALUES  
      ('1', 'эксперимент', '2025-02-21', '54 seconds'),  
      ('2', 'несчастный случай', '1999-01-23', '4 minutes 56 seconds'),  
      ('3', 'убийство', '2003-04-12', '3 hours 12 minutes 14 seconds');  
INSERT INTO outcome VALUES  
      ('1', 'здоров', TRUE, NULL, 'рекомпрессия'),  
      ('2', 'травмирован', TRUE, 'удалось спасти, но остался частично парализованным из-  
за воздушной эмболии', null),  
      ('3', 'мёртв', FALSE, 'летальный исход', NULL);  
END;
```

Триггер и связанная с ним функция

Данный триггер вызывается при каждом добавлении или обновлении данных в `medical_data`. Функция внутри триггера проверяет, что полученный `normal_pressure` соответствует регулярному выражению

```
CREATE FUNCTION validate_pressure()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.normal_pressure !~* '^[1-9][0-9]{0,3}/[1-9][0-9]{0,2}$'
    THEN
        RAISE EXCEPTION 'Неверный формат normal_pressure';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_pressure BEFORE INSERT OR UPDATE ON medical_data
FOR EACH ROW EXECUTE FUNCTION validate_pressure();
```

Вывод

При выполнении лабораторной работы я узнал, что такое нормализация и денормализация. Я научился определять функциональные зависимости моделей и анализировать её с помощью нормальных форм. Также я познакомился с языком PL/pgSQL.