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 P3106  
Мельник Фёдор Александрович

Проверил:

Вербовой Александр Александрович,

Преподаватель-практик ФПИиКТ

Санкт-Петербург, 2025

Оглавление

[Текст задания 3](#_Toc193478255)

[Выполнение 4](#_Toc193478256)

[Функциональные зависимости 4](#_Toc193478257)

[Нормальные формы 5](#_Toc193478258)

[1NF 5](#_Toc193478259)

[2NF 5](#_Toc193478260)

[3NF 5](#_Toc193478261)

[BCNF 6](#_Toc193478262)

[Денормализация 6](#_Toc193478263)

[Триггер и связанная с ним функция 6](#_Toc193478264)

[Вывод 7](#_Toc193478265)

# Текст задания

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, диаграмма, число

Контент, сгенерированный ИИ, может содержать ошибки.

# Выполнение

## Функциональные зависимости

person: person\_id → person\_name, person\_id, medical\_id

medical\_data: medical\_id → person\_health\_status, normal\_pressure

session\_in\_wacuum: session\_id → person\_id, session\_date, session\_interval, reason

outcome: session\_id → description, is\_survived, person\_status, requirements

person\_locations: person\_id → 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: person(person\_id → medical\_id) и medical\_data(medical\_id → person\_health\_status)

Исправим, убрав зависимость person(person\_id → medical\_id) и добавив medical\_data (medical\_id → person\_id)

Новая схема:  
Изображение выглядит как текст, снимок экрана, диаграмма, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

### BCNF

BCNF выполняется, так как отношения находятся в 3NF (после изменений) и для каждой функциональной зависимости X → 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'),

('Дроджия', '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 выполнение 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.