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

Факультет программной инженерии и компьютерной техники

Лабораторная работа по базам данных №3

Вариант № 31071

Выполнил:
Студент группы P3106
Мельник Фёдор Александрович
Проверил:
Вербовой Александр Александрович,
Преподаватель-практик ФПИиКТ

Санкт-Петербург, 2025

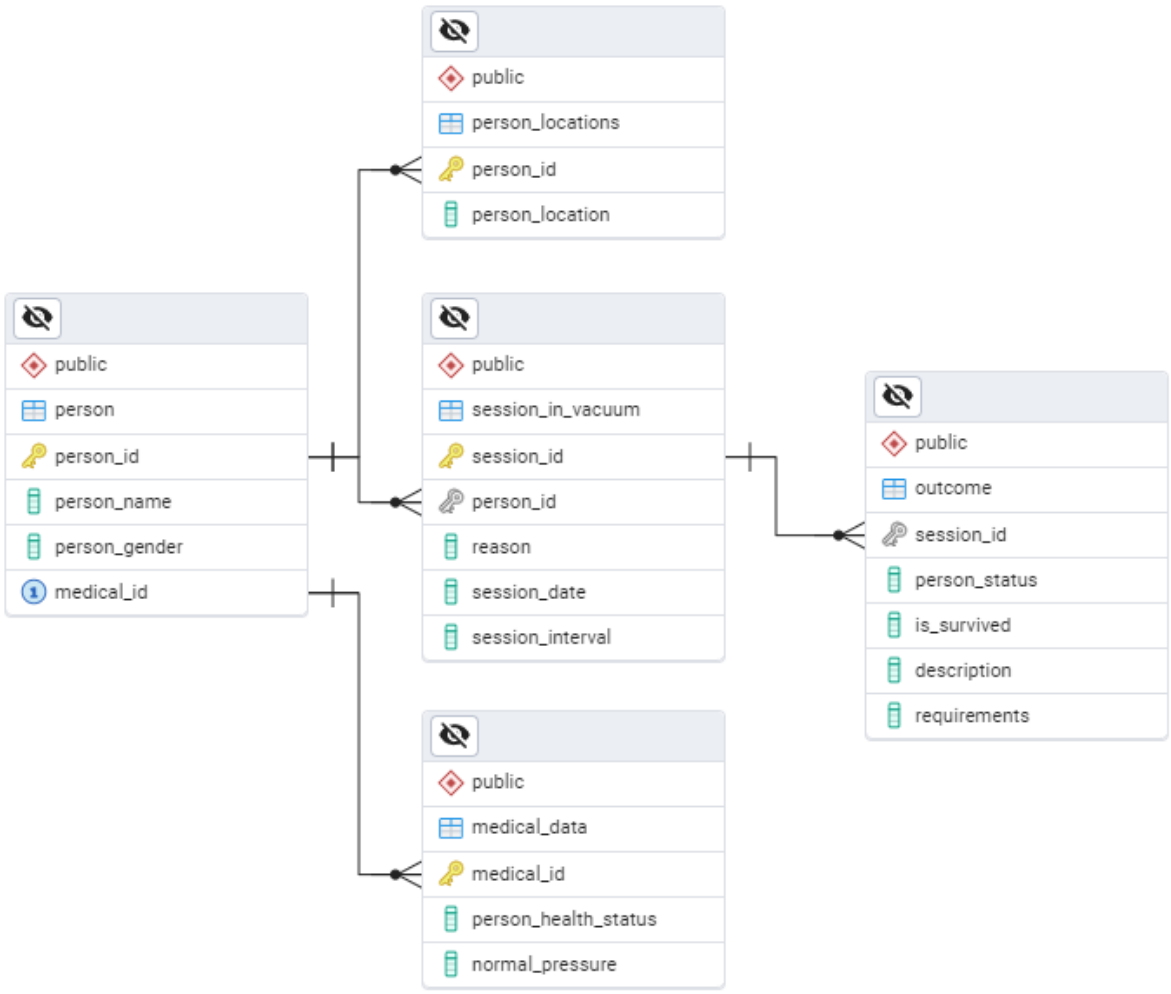
Оглавление

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 основеNF (как минимум).
- Опишите изменения в функциональных зависимостях, произошедшие после преобразования в 3NF (как минимум). Постройте схему на основеNF;
- 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 → 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

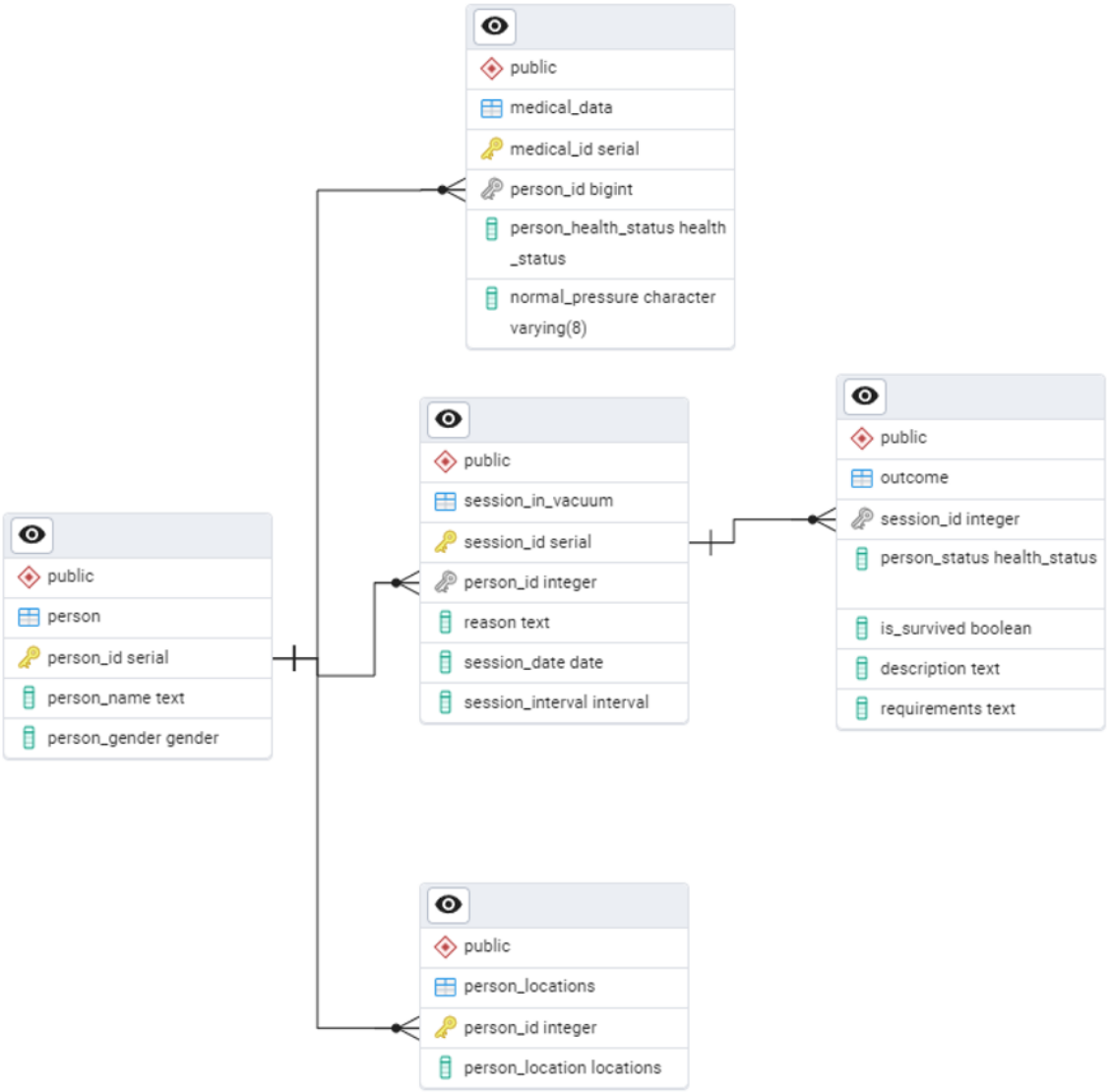
Вторая нормальная форма выполняется, так как отношения находятся в 1NF и все атрибуты зависят от всего первичного ключа, а не его части

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:



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 TABLE IF NOT EXISTS dublicates
(
  id SERIAL PRIMARY KEY,
  table_first TEXT,
  table_second TEXT,
  dublicate_first TEXT,
  dublicate_second TEXT,
  tables_key TEXT
);

INSERT INTO dublicates (table_first, table_second, dublicate_first, dublicate_second, tables_key)
VALUES
('person', 'medical_data', 'person_health_status', 'person_health_status', 'person_id');

CREATE FUNCTION update_dublicates()
RETURNS TRIGGER AS $$
DECLARE
  table_second_name TEXT;
  dublicate_second_name TEXT;
  dublicate_second_value TEXT;
  tables_key_name TEXT;
  tables_key_value TEXT;
  tables_key_type TEXT;
  dublicates_type TEXT;
BEGIN
  SELECT table_second, dublicate_second, tables_key
```

```
  INTO table_second_name, dublicate_second_name, tables_key_name
  FROM dublicates
  WHERE table_first = TG_TABLE_NAME AND dublicate_first = TG_ARGV[0];

  IF NOT FOUND THEN
    RETURN NEW;
  END IF;

  EXECUTE format('SELECT ($1).%I FROM (SELECT $1 AS new_record) AS subquery',
tables_key_name)
  INTO tables_key_value
  USING NEW;

  EXECUTE format('SELECT ($1).%I FROM (SELECT $1 AS new_record) AS subquery',
dublicate_second_name)
  INTO dublicate_second_value
  USING NEW;

  EXECUTE format('SELECT pg_typeof(%I) FROM %I LIMIT 1', dublicate_second_name,
table_second_name)
    INTO dublicates_type;

  EXECUTE format('SELECT pg_typeof(%I) FROM %I LIMIT 1', tables_key_name,
table_second_name)
    INTO tables_key_type;

  EXECUTE format('UPDATE %I SET %I = $1::%s WHERE %I = $2::%s', table_second_name,
dublicate_second_name, dublicates_type, tables_key_name, tables_key_type)
  USING dublicate_second_value, tables_key_value;

  RETURN NEW;
END;
```

```
$$ LANGUAGE plpgsql;


CREATE TRIGGER sync_medical_data
AFTER UPDATE OF person_health_status ON medical_data
FOR EACH ROW
EXECUTE FUNCTION update_dublicates('person_health_status');


CREATE TRIGGER sync_person
AFTER UPDATE OF person_health_status ON person
FOR EACH ROW
EXECUTE FUNCTION update_dublicates('person_health_status');
```

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.