Колледж Автономной некоммерческой образовательной организации

высшего образования

«Научно-технологический университет «Сириус»

Пояснительная записка по результату выполнения работы в рамках курса

«Технология разработки баз данных»

Тема: «Автоматизированная система для медецинских учреждений»

Работу подготовил:	
Студент группы	*К0709-22*
	Аргун В. Г.
Проверил:	
преподаватель	Картавых Е.В.

Сириус 2025

Оглавление

1 Введение 3

2 Описание предметной области 3

3 Проектирование БД 4

3.1 Первичный список отношений 4

3.2 Анализ отношения «patients» 4

3.2.1 Проверка на соответствие 1НФ 4

3.2.2 Проверка на соответствие 2НФ 5

3.2.3 Проверка на соответствие 3НФ 5

3.3 Анализ отношения «doctors» 5

3.3.1 Проверка на соответствие 1НФ 5

3.3.2 Проверка на соответствие 2НФ 5

3.3.3 Проверка на соответствие 3НФ 5

3.4 Анализ отношения «appointments» 5

3.4.1 Проверка на соответствие 1НФ 5

3.4.2 Проверка на соответствие 2НФ 6

3.4.3 Проверка на соответствие 3НФ 6

3.5 Анализ отношения «doctor_specialties» 6

3.5.1 Проверка на соответствие 1НФ 6

3.5.2 Проверка на соответствие 2НФ 6

3.5.3 Проверка на соответствие 3НФ 6

3.6 Анализ отношения «diagnoses» 6

3.6.1 Проверка на соответствие 1НФ 6

3.6.2 Проверка на соответствие 2НФ 7

3.6.3 Проверка на соответствие 3НФ 7

4 Таблицы базы данных 7

5 Связи 9

6 Пример данных 10

7 Заключение 11

1 Введение

В данной работе разработана структура базы данных для хранения информации медицинского учреждения. База данных содержит сущности для учёта пациентов, врачей, приёмов, специальностей врача и результатов диагностики. Цель проекта – обеспечить целостность данных, устранить избыточность посредством нормализации (приведение отношений к 3НФ) и повысить производительность системы за счёт использования UUID, индексов и строгих ограничений.

2 Описание предметной области

При первоначальном анализе предметной области были выявлены следующие сущности:

- **Пациент (patients):** человек, получающий медицинскую помощь.
    - Имя (текст)
    - Фамилия (текст)
    - Дата рождения (дата)
    - Пол (цифра)
    - Контактный телефон (текст)
    - Адрес (текст)
- **Врач (doctors):** медицинский работник, оказывающий помощь пациентам.
    - Имя (текст)
    - Фамилия (текст)
    - Контактные данные (текст)
    - Email (текст)
    - Дата приёма на работу (дата)
- **Приём (appointments):** запись о визите пациента к врачу.
    - Дата (дата)
    - Время (время)
    - Статус (текст)
- **Диагноз (diagnoses):** сущность для хранения результатов диагностики.
    - Код диагноза (текст)
    - Описание (текст)
    - Связанный приём (идентификатор)
- **Специальности врача (doctor_specialties):** перечень специальностей, которыми обладает врач.
    - Специальность (текст)

3 Проектирование БД

3.1 Первичный список отношений

Исходя из описания предметной области, сформирован следующий первичный список отношений:

1. **patients**
   a. patient_id
   b. first_name
   c. last_name
   d. date_of_birth
   e. gender
   f. phone
   g. address
2. **doctors**
   a. doctor_id
   b. first_name
   c. last_name
   d. phone
   e. email
   f. hire_date
3. **appointments**
   a. appointment_id
   b. patient_id
   c. doctor_id
   d. appointment_datetime
   e. status
4. **doctor_specialties**
   a. doctor_specialty_id
   b. doctor_id
   c. specialty
5. **diagnoses**
   a. diagnosis_id
   b. appointment_id
   c. diagnosis_code
   d. description

3.2 Анализ отношения «patients»

3.2.1 Проверка на соответствие 1НФ

Начальное состояние:
В исходной версии поле *phone* могло содержать несколько номеров (например, «+7 123..., +7 456...»), а *gender*задавалось как строка, что допускало произвольный ввод.

Действия для приведения к 1НФ:

- Преобразовать поле *phone* к хранению одного номера.

- Поле *gender* изменить на числовой тип (SMALLINT) с ограничением (0 – неопределён, 1 – Male, 2 – Female).

3.2.2 Проверка на соответствие 2НФ

Первичный ключ – *patient_id*. Все неключевые атрибуты (first_name, last_name, date_of_birth, gender, phone, address) зависят полностью от него. Так как ключ одиночный, частичных зависимостей нет.

3.2.3 Проверка на соответствие 3НФ

Отсутствуют транзитивные зависимости – каждый атрибут напрямую зависит от *patient_id*, и нет атрибутов, зависящих друг от друга. Отношение удовлетворяет 3НФ.

3.3 Анализ отношения «doctors»

3.3.1 Проверка на соответствие 1НФ

Начальное состояние:
Первоначально предполагалось хранить список специальностей врача в одном поле (например, «Cardiology, Neurology»), что нарушало атомарность данных.

Действия для приведения к 1НФ:

- Исключить поле специальностей из *doctors* и хранить только базовые данные: first_name, last_name, phone, email, hire_date.
- Вынести специальности во внешнюю таблицу (doctor_specialties).

3.3.2 Проверка на соответствие 2НФ

Первичный ключ – *doctor_id*. Все остальные поля (first_name, last_name, phone, email, hire_date) зависят полностью от него. Нет частичных зависимостей.

3.3.3 Проверка на соответствие 3НФ

Нет транзитивных зависимостей: все поля напрямую зависят от *doctor_id*. Отношение удовлетворяет 3НФ.

3.4 Анализ отношения «appointments»

3.4.1 Проверка на соответствие 1НФ

Начальное состояние:
Изначально в таблице *appointments* присутствовало поле *reason*, которое могло содержать составные данные (например, «Боль в груди, кашель»), а *status* задавался как произвольная строка.

Действия для приведения к 1НФ:

- Удалить поле *reason* (не имело однозначной смысловой нагрузки).
- Преобразовать поле *status* в SMALLINT с набором значений (0 – запланирован, 1 – завершён, 2 – отменён, 3 – перенесён).

3.4.2 Проверка на соответствие 2НФ

Первичный ключ – *appointment_id*. Все поля (patient_id, doctor_id, appointment_datetime, status) зависят целиком от него, поскольку ключ одиночный.

3.4.3 Проверка на соответствие 3НФ

Отсутствуют транзитивные зависимости: ни один неключевой атрибут не зависит от другого неключевого атрибута. Отношение удовлетворяет 3НФ.

3.5 Анализ отношения «doctor_specialties»

3.5.1 Проверка на соответствие 1НФ

Начальное состояние:
Предполагалось хранить несколько специальностей врача в одной строке. Это нарушало принцип атомарности.

Действия для приведения к 1НФ:

- Создать отдельную таблицу *doctor_specialties*, где каждая запись содержит одну специальность, связанную с конкретным врачом (doctor_id).

3.5.2 Проверка на соответствие 2НФ

Первичный ключ – *doctor_specialty_id*. Поля (doctor_id, specialty) зависят полностью от него. Нет частичных зависимостей.

3.5.3 Проверка на соответствие 3НФ

Нет транзитивных зависимостей: specialty напрямую зависит от doctor_specialty_id. Отношение удовлетворяет 3НФ.

3.6 Анализ отношения «diagnoses»

3.6.1 Проверка на соответствие 1НФ

Начальное состояние:
Изначально планировалось хранить диагнозы в таблице *appointments*, что затрудняло хранение нескольких диагнозов для одного приёма и вело к избыточности.

Действия для приведения к 1НФ:

- Вынести диагнозы в отдельную таблицу *diagnoses*, где поля (diagnosis_code, description) являются атомарными, а appointment_id – внешний ключ на *appointments*.

3.6.2 Проверка на соответствие 2НФ

Первичный ключ – *diagnosis_id*. Поля (appointment_id, diagnosis_code, description) зависят полностью от него. Нет частичных зависимостей.

3.6.3 Проверка на соответствие 3НФ

Отсутствуют транзитивные зависимости: все неключевые атрибуты напрямую зависят от diagnosis_id. Отношение удовлетворяет 3НФ.

4 Таблицы базы данных

Ниже приведён скрипт для создания всех таблиц в базе данных:

```
CREATE EXTENSION IF NOT EXISTS "uuid-ossp";


CREATE TABLE patients (
    patient_id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    date_of_birth DATE NOT NULL,
    gender SMALLINT NOT NULL CHECK (gender IN (0, 1, 2)),
    phone VARCHAR(20),
    address VARCHAR(255)
);


CREATE TABLE doctors (
    doctor_id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    phone VARCHAR(20),
    email VARCHAR(100),
    hire_date DATE
);
```

```
CREATE TABLE appointments (
    appointment_id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    patient_id UUID NOT NULL,
    doctor_id UUID NOT NULL,
    appointment_datetime TIMESTAMP NOT NULL,
    status SMALLINT NOT NULL CHECK (status IN (0, 1, 2, 3))
);

CREATE TABLE doctor_specialties (
    doctor_specialty_id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    doctor_id UUID NOT NULL,
    specialty VARCHAR(100) NOT NULL
);

CREATE TABLE diagnoses (
    diagnosis_id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    appointment_id UUID NOT NULL,
    diagnosis_code VARCHAR(20) NOT NULL,
    description TEXT
);
```

5 Связи

В базе данных имеются следующие связи между отношениями:

1. **Связь между пациентом и приёмом**
   Это связь один ко многим. Один пациент может иметь несколько приёмов, а один приём связан только с одним пациентом. Связь реализуется через внешний ключ *patient_id* в таблице appointments:

   ```
   ALTER TABLE appointments

   ADD FOREIGN KEY (patient_id) REFERENCES patients(patient_id);
   ```

2. **Связь между врачом и приёмом**
   Это связь один ко многим. Один врач может вести несколько приёмов, а каждый приём проводится одним врачом. Связь реализуется через внешний ключ *doctor_id* в таблице appointments:

   ```
   ALTER TABLE appointments

   ADD FOREIGN KEY (doctor_id) REFERENCES doctors(doctor_id);
   ```

3. **Связь между врачом и специальностью врача**
   Это связь один ко многим. Один врач может обладать несколькими специальностями, а одна запись в таблице doctor_specialties относится к конкретному врачу. Реализуется через внешний ключ *doctor_id* в таблице doctor_specialties:

   ```
   ALTER TABLE doctor_specialties

   ADD FOREIGN KEY (doctor_id) REFERENCES doctors(doctor_id);
   ```

4. **Связь между приёмом и диагнозом**
   Это связь один ко многим. Один приём может иметь несколько диагнозов, а один диагноз связан только с одним приёмом. Реализуется через внешний ключ *appointment_id* в таблице diagnoses:

   ```
   ALTER TABLE diagnoses

   ADD FOREIGN KEY (appointment_id) REFERENCES appointments(appointment_id);
   ```

Таким образом, все таблицы участвуют в связях. Если бы остались отношения, не связанные с другими таблицами, это требовало бы отдельного обоснования.

6 Пример данных

1. **Пример данных для таблицы patients**

```
INSERT INTO patients (patient_id, first_name, last_name, date_of_birth,
gender, phone, address)

VALUES

('11111111-1111-1111-1111-111111111111', 'Sergey', 'Sidorov', '1980-05-15',
1, '+7 123 456-78-90', 'ул. Ленина, 10'),

('22222222-2222-2222-2222-222222222222', 'Elena', 'Kuznetsova', '1990-08-22',
2, '+7 987 654-32-10', 'ул. Пушкина, 5');
```

2. **Пример данных для таблицы doctors**

```
INSERT INTO doctors (doctor_id, first_name, last_name, phone, email,
hire_date)

VALUES

('aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa', 'Ivan', 'Petrov', '+7 111 222-33-
44', 'ivan.petrov@example.com', '2010-01-15'),

('bbbbbbbb-bbbb-bbbb-bbbb-bbbbbbbbbbbb', 'Anna', 'Smirnova', '+7 222 333-44-
55', 'anna.smirnova@example.com', '2012-06-20');
```

3. **Пример данных для таблицы appointments**

```
INSERT INTO appointments (appointment_id, patient_id, doctor_id,
appointment_datetime, status)

VALUES

('77777777-7777-7777-7777-777777777777', '11111111-1111-1111-1111-
111111111111', 'aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa', '2025-03-20 09:00:00',
0),

('88888888-8888-8888-8888-888888888888', '22222222-2222-2222-2222-
222222222222', 'bbbbbbbb-bbbb-bbbb-bbbb-bbbbbbbbbbbb', '2025-03-21 10:30:00',
1);
```

4. **Пример данных для таблицы doctor_specialties**

```
INSERT INTO doctor_specialties (doctor_specialty_id, doctor_id, specialty)

VALUES

('44444444-4444-4444-4444-444444444444', 'aaaaaaaa-aaaa-aaaa-aaaa-
aaaaaaaaaaaa', 'Cardiology'),

('55555555-5555-5555-5555-555555555555', 'bbbbbbbb-bbbb-bbbb-bbbb-
bbbbbbbbbbbb', 'Neurology');
```

5. **Пример данных для таблицы diagnoses**

```
INSERT INTO diagnoses (diagnosis_id, appointment_id, diagnosis_code,
description)

VALUES

('77777777-7777-7777-7777-777777777779', '77777777-7777-7777-7777-
777777777777', 'I20', 'Angina pectoris'),

('88888888-8888-8888-8888-888888888889', '88888888-8888-8888-8888-
888888888888', 'G44', 'Migraine');
```

7 Заключение

Описана и создана структура базы данных для хранения информации медицинского учреждения. База данных содержит **5** отношений. Все отношения нормализованы до третьей нормальной формы. Между отношениями установлены и реализованы связи с использованием ограничения **FOREIGN KEY**. Всего таких связей **4**. В разделе 6 приведён пример заполнения данных.

В процессе выполнения работы были получены навыки по нормализации (1НФ, 2НФ, 3НФ), проектированию таблиц с учётом требований предметной области и настройке ограничений целостности (CHECK, FOREIGN KEY). Это решение обеспечивает целостность и масштабируемость, а при необходимости легко расширяется новыми сущностями.