



# Курсовая работа

По учебной дисциплине: “Базы данных и сетевые технологии”

**Тема:** “Проектирование базы данных для частной медицинской клиники”

Работа выполнена: Жмаевым Романом  
Студент СПбГУ, ф. ПМ-ПУ, гр. 23.Б02-ПУ

Г. Санкт-Петербург, 2024

# Оглавление

Введение .....	3
Описание базы данных .....	4
Схема .....	5
Запросы .....	6
Простые .....	6
Средние .....	7
Сложные .....	9
Заключение.....	11

## Введение

В данной курсовой работе описывается процесс создания базы данных для модели частной медицинской клиники (её работы, персонала и услугах).

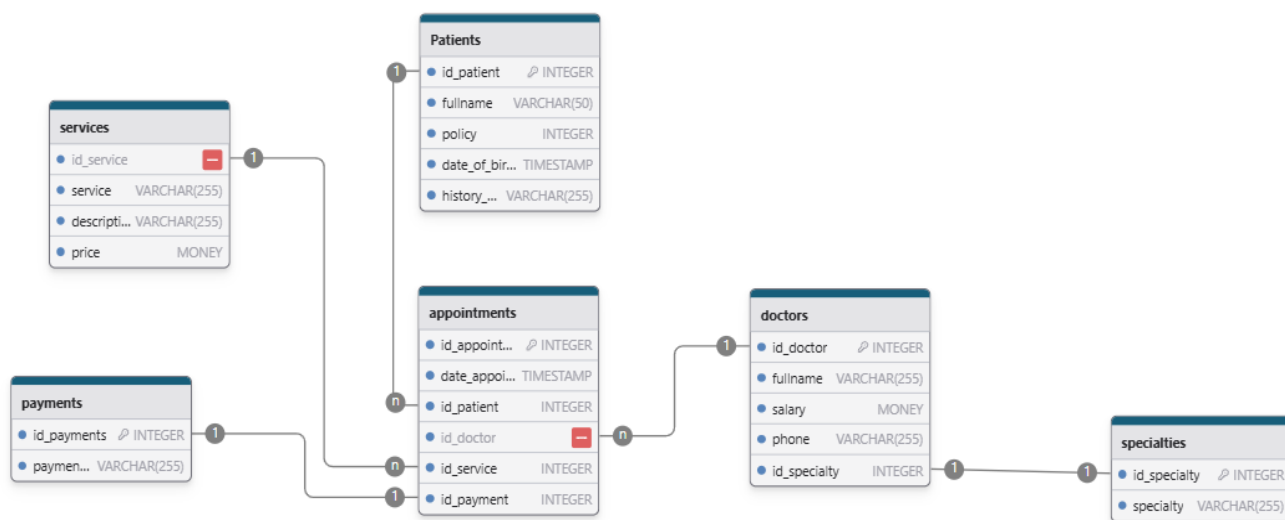
**Цель:** получить практические навыки создания базы данных с использованием языка запросов SQL и СУБД PostgreSQL, обращаясь к полученным на лекциях знаниям.

### **Задачи:**

- Описать схему базы данных для частной медицинской клиники;
- Наполнить полученные таблицы данными для широты выборки;
- Согласовать данные между собой и наложить определенные ограничения на конкретные столбцы;
- Создание запросов разной сложности для взаимодействия с данными.

## Описание базы данных

База данных состоит из 6 таблиц: appointments, doctors, patients, payments, services, specialties (рис. 1)



(рис. 1 “Схема базы данных”)

## Схема

### 1. **services** (услуги)

- **id\_service** (INTEGER): Уникальный идентификатор услуги.
- **service** (VARCHAR(255)): Название услуги.
- **descript\_** (VARCHAR(255)): Описание услуги.
- **price** (MONEY): Стоимость услуги.

### 2. **payments** (платежи)

- **id\_payments** (INTEGER): Уникальный идентификатор платежа.
- **payment\_** (VARCHAR(255)): Детали платежа.

### 3. **patients** (пациенты)

- **id\_patient** (INTEGER): Уникальный идентификатор пациента.
- **fullname** (VARCHAR(50)): Полное имя пациента.
- **policy** (INTEGER): Полис или номер документа пациента.
- **date\_of\_bir\_** (TIMESTAMP): Дата рождения пациента.
- **history\_** (VARCHAR(255)): История болезни пациента.

### 4. **appointments** (записи на прием)

- **id\_appoint\_** (INTEGER): Уникальный идентификатор записи.
- **date\_appoint\_** (TIMESTAMP): Дата записи на прием.
- **id\_patient** (INTEGER): Идентификатор пациента (внешний ключ, связывается с **id\_patient** таблицы **patients**).
- **id\_doctor** (INTEGER): Идентификатор врача (внешний ключ, связывается с **id\_doctor** таблицы **doctors**).
- **id\_service** (INTEGER): Идентификатор услуги (внешний ключ, связывается с **id\_service** таблицы **services**).
- **id\_payment** (INTEGER): Идентификатор платежа (внешний ключ, связывается с **id\_payments** таблицы **payments**).

### 5. **doctors** (врачи)

- **id\_doctor** (INTEGER): Уникальный идентификатор врача.
- **fullname** (VARCHAR(255)): Полное имя врача.
- **salary** (MONEY): Зарплата врача.
- **phone** (VARCHAR(255)): Контактный телефон врача.

- `id_specialty` (INTEGER): Идентификатор специальности (внешний ключ, связывается с `id_specialty` таблицы `specialties`).

## 6. **specialties** (специальности)

- `id_specialty` (INTEGER): Уникальный идентификатор специальности.
- `specialty` (VARCHAR(255)): Название специальности.


### Связи между таблицами:

- **services** связана с **appointments** по полю `id_service`.
- **payments** связана с **appointments** по полю `id_payment`.
- **patients** связана с **appointments** по полю `id_patient`.
- **doctors** связана с **appointments** по полю `id_doctor`.
- **specialties** связана с **doctors** по полю `id_specialty`.

## Запросы

### Простые

sql

 Копировать код

```
SELECT date_appointment
FROM appointments
WHERE date_appointment BETWEEN '2023-01-01 00:00:00' AND '2024-12-30 00:00:00';
```

Этот запрос выбирает даты записей на прием (`date_appointment`) из таблицы `appointments`, которые находятся в диапазоне с 1 января 2023 года до 30 декабря 2024 года (включительно). Список всех дат приемов, которые соответствуют указанному временному периоду.


sql

 Копировать код

```
SELECT service, price
FROM services
WHERE price > 30000
ORDER BY price DESC;
```

Этот запрос выбирает названия услуг (`service`) и их стоимость (`price`) из таблицы `services`, где цена превышает **30,000**. Результат отсортирован по цене в порядке убывания (самые дорогие услуги будут показаны первыми). Список услуг с их стоимостью, где цена выше 30,000, отсортированный по убыванию цены.

sql

 Копировать код

```
SELECT id_payment as only_card
FROM payments
WHERE payment_method IN ('credit card', 'debit card');
```

Этот запрос выбирает идентификаторы платежей (id\_payment) из таблицы payments, где метод оплаты указан как **"кредитная карта"** (credit card) или **"дебетовая карта"** (debit card).

Результаты выводятся с псевдонимом only\_card (переименованное имя столбца). Список идентификаторов платежей, которые были выполнены с использованием кредитной или дебетовой карты.

sql

 Копировать код

```
SELECT id_patient, fullname, history_of_visiting
FROM patients
WHERE history_of_visiting LIKE 'Рак%'
ORDER BY history_of_visiting ASC;
```

Этот запрос выбирает идентификаторы пациентов (id\_patient), их полные имена (fullname) и историю посещений (history\_of\_visiting) из таблицы patients, где история посещений начинается с слова **"Рак"** (например, "Рак легких", "Рак кожи").

Результаты отсортированы в алфавитном порядке по полю history\_of\_visiting. Список пациентов, у которых в истории посещений есть записи, начинающиеся с «Рак», отсортированный по алфавиту в порядке возрастания.

## Средние

sql

 Копировать код

```
SELECT a.date_appointment, p.fullname, p.policy
FROM appointments a
JOIN patients p ON a.id_patient = p.id_patient
WHERE p.history_of_visiting = 'Здоров'
ORDER BY fullname DESC;
```

Этот запрос соединяет таблицы appointments (записи на прием) и patients (пациенты) по совпадению их идентификаторов (id\_patient). Выбирает дату записи на прием (date\_appointment), полное имя пациента (fullname) и его полис (policy).

Учитывает только тех пациентов, у которых история посещений равна "Здоров". Сортирует результат по имени пациента в порядке убывания (сначала фамилии, стоящие ближе к концу алфавита). Список записей на прием для пациентов с историей "Здоров", с их датами приема, именами и номерами полиса, отсортированный по именам в обратном алфавитном порядке.

sql  Копировать код

```
SELECT s.speciality, COUNT(d.fullname) as doctors
FROM doctors d
JOIN specialties s ON d.id_speciality = s.id_speciality
GROUP BY s.speciality
HAVING COUNT(s.speciality) >= 2
ORDER BY doctors DESC;
```

Этот запрос соединяет таблицы doctors (врачи) и specialties (специальности) по полю id\_speciality. Группирует данные по специальностям (speciality). Считает количество врачей (fullname), относящихся к каждой специальности. Отбирает только те специальности, у которых 2 или более врача (условие HAVING COUNT(s.speciality) >= 2). Сортирует результаты по количеству врачей в порядке убывания. Список специальностей с количеством врачей, где каждая специальность представлена как минимум двумя врачами, отсортированный по количеству врачей (от большего к меньшему).

sql  Копировать код

```
SELECT payment_method, COUNT(id_payment) as count_of_methods
FROM payments
GROUP BY payment_method
HAVING COUNT(id_payment) > 2;
```

Этот запрос выбирает методы оплаты (payment\_method) и подсчитывает количество транзакций для каждого метода (COUNT(id\_payment)). Группирует данные по методам оплаты (GROUP BY payment\_method). Отбирает только те методы, которые использовались более двух раз (условие HAVING COUNT(id\_payment) > 2). Список методов оплаты, которые применялись более двух раз, с указанием количества транзакций для каждого метода.



## Сложные

```
sql Копировать код  
  
SELECT (SELECT service  
        FROM services s  
        WHERE s.id_service = a.id_service  
        ), COUNT(a.id_doctor) as count_of_doctors  
FROM appointments a  
JOIN doctors d ON a.id_doctor = d.id_doctor  
WHERE d.salary > 50000 AND a.date_appointment BETWEEN '2021-01-01 00:00:00' AND '2022-12-31 00:00:00'  
GROUP BY a.id_service  
HAVING COUNT(a.id_doctor) >= 3  
ORDER BY COUNT(a.id_doctor) DESC;
```


Этот запрос для каждой услуги (service) из таблицы services (определяемой через подзапрос, связанный с id\_service из appointments), вычисляет количество врачей (COUNT(a.id\_doctor)), которые проводили приемы, связанные с этой услугой. Соединяет таблицы appointments и doctors, проверяя, чтобы зарплата врача (salary) превышала 50,000. Учитывает только записи на приемы, которые были проведены в период с 1 января 2021 года по 30 декабря 2022 года. Группирует данные по услугам (id\_service) и отбирает только те услуги, где количество врачей, проводивших приемы, составляет 3 или более. Сортирует результаты по количеству врачей в порядке убывания. Список услуг и количества врачей, которые проводили приемы по этим услугам (с зарплатой более 50,000) в заданный период, отсортированный по количеству врачей.

```
sql Копировать код  
  
SELECT a.date_appointment, pay.payment_method, p.fullname  
FROM appointments a  
JOIN payments pay ON a.id_payment = pay.id_payment  
LEFT JOIN patients p ON p.id_patient = a.id_patient  
WHERE a.date_appointment BETWEEN '2020-01-01 00:00:00' AND '2020-12-30 00:00:00'  
ORDER BY a.date_appointment DESC;
```

Этот запрос соединяет таблицу appointments с таблицей payments через id\_payment, чтобы получить информацию о методе оплаты (payment\_method). Выполняет левое соединение с таблицей patients по id\_patient, чтобы включить полное имя пациента (fullname), если таковой существует. Отбирает записи на приемы (date\_appointment), которые были проведены в период с **1 января 2020 года по 30 декабря 2020 года**. Сортирует результаты по дате записи на прием в порядке убывания (сначала более поздние даты). Список дат приемов, методов оплаты и имен пациентов (если данные

пациента доступны) за 2020 год, отсортированный от самых поздних дат к более ранним.

sql

 Копировать код

```
SELECT (SELECT p.fullname
        FROM patients P
        WHERE a.id_patient = p.id_patient
      ), a.id_patient, SUM(s.price) as sum_price
FROM appointments a
JOIN services s ON a.id_service = s.id_service
WHERE s.price > 30000
GROUP BY a.id_patient
HAVING COUNT(s.id_service) > 2
ORDER BY sum_price DESC;
```

Этот запрос выполняет подзапрос для получения полного имени пациента (fullname) из таблицы patients, связанного с id\_patient из таблицы appointments. Подсчитывает суммарную стоимость услуг (SUM(s.price)), связанных с каждым пациентом, где стоимость услуг превышает 30,000. Группирует данные по пациентам (id\_patient). Отбирает только тех пациентов, которые пользовались услугами более двух раз (условие HAVING COUNT(s.id\_service) > 2). Сортирует результаты по суммарной стоимости услуг в порядке убывания. Список пациентов, идентификаторов и суммарной стоимости услуг, где каждая услуга стоила более 30,000, а пациент пользовался такими услугами более двух раз. Результаты отсортированы по суммарной стоимости услуг (от большей к меньшей).

## Заключение

Поставленные задачи были выполнены, а конечным продуктом является база данных для медицинской клиники также были усвоены практические методы обработки данных с помощью SQL и PostgreSQL. Созданные таблицы, связи между ними и написанные SQL-запросы позволяют эффективно обрабатывать информацию, предоставляя пользователям удобный доступ к данным.