

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение  
высшего образования «Санкт-Петербургский политехнический университет  
Петра Великого»

Институт компьютерных наук и кибербезопасности

Высшая школа технологий и искусственного интеллекта

Направление: 02.03.01 Математика и компьютерные науки

Отчет о выполнении курсовой работы

"Реализация базы данных для предметной области «Салон  
маникюра»"

по дисциплине «Теоретические основы баз данных»

Студент,  
группа 5130201/30002

Преподаватель,  
к.т.н., доц., доц.

Михайлова А.А

Попов С.Г

«3» 05 2025г.

Санкт-Петербург, 2025

# Реферат

Предметной областью данной работы является салон маникюра. Основная проблема заключается в отсутствии систематизированного хранения информации о клиентах, записях, услугах и сотрудниках, что усложняет процесс управления салоном. В рамках работы была разработана база данных, хранящая информацию о сущностях салона и их связях. Были сформированы и выполнены 8 запросов, 2 из которых включают в себя второй вариант.

Для выполнения работы использовалсяialect MySQL, среда программирования PyCharm, Python версии 3.1, командная строка и MySQL Workbench 8.0.

Полученная база данных может быть использована в маникюрных салонах для автоматизации учета клиентов, оптимизации графика работы мастеров, анализа востребованности услуг и улучшения качества обслуживания.

# Содержание

|  |           |
|--|-----------|
| <b>Реферат</b>   | <b>2</b>  |
| <b>Введение</b>  | <b>4</b>  |
| <b>1 Постановка задачи</b>   | <b>5</b>  |
| <b>2 Аналитика</b>   | <b>6</b>  |
| 2.1 Описание предметной области . . . . .  | 6         |
| 2.2 Цели функционирования базы данных . . . . .                                      | 9         |
| 2.3 ER - диаграмма . . . . .   | 9         |
| 2.4 Сущности и их атрибуты . . . . .   | 9         |
| 2.5 Перечень ролей . . . . .   | 10        |
| 2.6 Схема объектов базы данных . . . . .   | 10        |
| <b>3 Проектирование</b>  | <b>11</b> |
| 3.1 Лингвистическое описание . . . . .   | 11        |
| 3.2 Таблицы метаданных . . . . .   | 11        |
| 3.3 Генерация записей . . . . .  | 18        |
| <b>4 Программирование</b>  | <b>19</b> |
| 4.1 Запрос 1 . . . . .   | 20        |
| 4.1.1 Вариант 1 . . . . .  | 20        |
| 4.1.2 Вариант 2 . . . . .  | 21        |
| 4.2 Запрос 2 . . . . .   | 22        |
| 4.3 Запрос 3 . . . . .   | 23        |
| 4.4 Запрос 4 . . . . .   | 24        |
| 4.5 Запрос 5 . . . . .   | 26        |
| 4.5.1 Вариант 1 . . . . .  | 26        |
| 4.5.2 Вариант 2 . . . . .  | 27        |
| 4.6 Запрос 6 . . . . .   | 29        |
| 4.7 Запрос 7 . . . . .   | 30        |
| 4.8 Запрос 8 . . . . .   | 31        |
| <b>Заключение</b>  | <b>34</b> |
| <b>Список использованных источников</b>  | <b>35</b> |
| <b>Приложение А. Исходный код программы генерации схемы базы данных</b>              | <b>36</b> |
| <b>Приложение В. Исходный код программы заполнения базы данных тестовыми данными</b> | <b>40</b> |

# **Введение**

База данных (БД) – совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных.

Система управления базами данных (СУБД) – комплекс программ, позволяющих создать базу данных и манипулировать данными (вставлять, обновлять, удалять и выбирать). Система обеспечивает безопасность, надёжность хранения и целостность данных, а также предоставляет средства для администрирования БД.

Разработка базы данных – это процесс проектирования, создания и настройки структуры для хранения, управления и обработки данных. Она необходима для организации больших объемов информации, чтобы данные были легко доступны, быстро обрабатывались и обеспечивали структурированное взаимодействие между пользователями и приложениями. Базы данных особенно важны для автоматизации процессов, аналитики и принятия решений, так как позволяют эффективно хранить, изменять, искать и анализировать сведения. Среди преимуществ разработки базы данных: упрощение управления данными, минимизация ошибок, повышение скорости работы с информацией, улучшение безопасности данных, а также возможности масштабирования и совместной работы в реальном времени. Корректно спроектированная база данных помогает упорядочить информацию и сократить затраты на её обработку.

# 1 Постановка задачи

Цель работы: реализация функционирующей реляционной базы данных для салона маникюра и реализация выданных запросов.

Задачи работы.

- Проанализировать предметную область.
- Создать ER-диаграмму в нотации Чена.
- Определить цели функционирования базы данных.
- Спроектировать схему базы данных.
- Перенести базу данных в «MySQL».
- Написать текст программы заполнения базы данных.
- Реализация запросов.
- Графическое представление результатов запросов.

## 2 Аналитика

### 2.1 Описание предметной области

Салон маникюра представляет собой пространство, где клиенты получают профессиональные услуги ухода за руками и ногтями. Клиент – лицо, которое обращается в салон за выполнением конкретных процедур. Клиентами могут быть как женщины, так и мужчины. Они взаимодействуют с сотрудниками салона: мастерами и администраторами. Мастера – это специалисты, которые выполняют услуги салона, а также подготовку рабочего места и инструментов, предоставление рекомендаций клиентам по уходу за кожей и ногтями. Они используют специализированное оборудование, расходные материалы и инструменты в работе. У каждого мастера фиксированное расписание. Администратор салона принимает заявки на запись от клиентов, в соответствии с расписанием мастеров, контролирует наличие материалов и возможные конфликтные ситуации, проводит оплату услуг через кассу или электронные платежи.

Услуга – конкретная процедура, которую предлагает салон. К основным услугам относятся: уход за ногтями (обработка кутикулы, придание ногтям нужной длины и формы, удаление мозолей и загрубевшей кожи, полировка ногтей и увлажнение кожи рук), снятие покрытия, покрытие гель лаком. Запись – фиксация времени и даты посещения клиента. Материалы и инструменты – средства, которые используют для выполнения услуги (гель лаки, пилки, ножницы, аппараты для маникюра).

Инструменты в маникюрных салонах сначала дезинфицируют, удаляя загрязнения и замачивая в дезинфицирующем растворе, после проводят предстерилизационную очистку с промыванием, механической очисткой и контролем качества, после чего инструменты стерилизуют в сухожаре и хранят в стерильных крафт-пакетах.

Кроме инструментов существуют расходные материалы, которые хранятся на складе. За их наличием следит администратор и, если нужно, пополняет запасы.

Клиент звонит в салон, администратор уточняет желаемую услугу, мастера, дату и время, предлагает свободное время для записи. В назначенный день и время клиента приветствует администратор, который сопровождает его к выбранному мастеру. После этого мастер приступает к выполнению процедуры, используя инструменты и материалы. После завершения клиент проверяет результат, дает обратную связь мастеру, оплачивает услугу через администратора и при желании записывается на следующую процедуру.

Если результат выполнения услуги клиенту не понравился, клиент может сразу же сообщить мастеру или администратору о проблеме. Администратор выясняет причину недовольства и предлагает решение: это может быть бесплатное исправление, предоставление скидки на следующие услуги

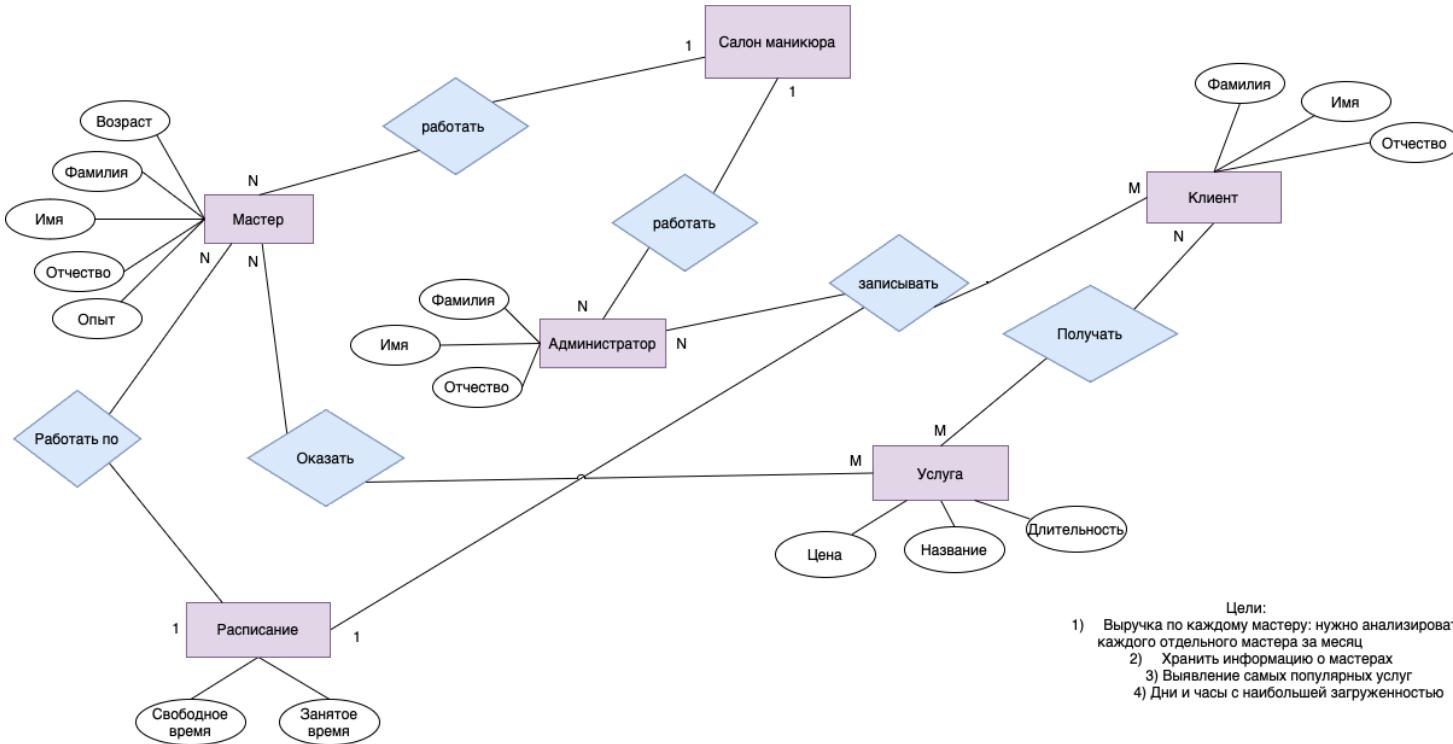
или частичный возврат денег.

Прейскурант — это документ, который содержит перечень услуг салона с указанием их стоимости. Составление прейскуранта основывается на нескольких факторах: себестоимость услуг, опыт мастера, класс салона, анализ цен конкурентов. Цены на услуги формируются на основе себестоимости, с учетом наценки. Уход за ногтями - 1000р., снятие покрытия – 500р., покрытие гель лаком – 2500р.

Набор инструментов и материалов для маникюра включает: одноразовые инструменты: перчатки, крафт-пакеты, безворсные салфетки), многоразовые инструменты: пилки, бафы, аппараты для маникюра, расходные материалы: антисептик, обезжириватель, гель-лак, база, топ, смягчающее средство для кутикулы.

Время на процедуру зависит от вида услуги. Уход за ногтями занимает около 1 часа, покрытие гель-лаком – 1,5 часа, снятие покрытия – 20 минут.

Оплата услуг проводится через кассу. Администратор салона выдает клиенту чек об оплате.



**Цели:**

- 1) Выручка по каждому мастеру: нужно анализировать доход каждого отдельного мастера за месяц
- 2) Хранить информацию о мастерах
- 3) Выявление самых популярных услуг
- 4) Дни и часы с наибольшей загруженностью

Мастер оказывает услугу;  
 Мастер работает по расписанию;  
 Мастер работает в салоне маникюра;  
 Администратор записывает клиента;  
 Администратор записывает по расписанию;  
 Администратор работает в салоне маникюра;  
 Клиент получает услугу.

Рис. 1: ER - диаграмма базы данных

## **2.2 Цели функционирования базы данных**

- Выручка по каждому мастеру: нужно анализировать доход каждого отдельного мастера за месяц
- Хранить информацию о мастерах
- Выявление самых популярных услуг
- Дни и часы с наибольшей загруженностью

## **2.3 ER - диаграмма**

- Мастер оказывает услугу;
- Мастер работает по расписанию;
- Мастер работает в салоне маникюра;
- Администратор записывает клиента;
- Администратор записывает по расписанию;
- Администратор работает в салоне маникюра;
- Клиент получает услугу.

## **2.4 Сущности и их атрибуты**

- Мастер – специалист, который выполняет услуги салона, а также подготовку рабочего места и инструментов, предоставление рекомендаций клиентам по уходу за кожей и ногтями.

Атрибуты: фамилия, имя, отчество.

- Администратор – принимает заявки на запись от клиентов, в соответствии с расписанием мастеров, контролирует наличие материалов и возможные конфликтные ситуации, проводит оплату услуг через кассу или электронные платежи.

Атрибуты: фамилия, имя, отчество.

- Клиент – лицо, которое обращается в салон за выполнением конкретных процедур.

Атрибуты: фамилия, имя, отчество.

- Услуга – конкретная процедура, которую предлагает салон.

Атрибуты: цена, название, длительность.

- Расписание – график, содержащий сведения о времени работы мастеров.
- Атрибуты: свободное время, занятое время.

## 2.5 Перечень ролей

- Мастер. Задача выполнить услугу салона.
- Администратор. Задача принять заявку на запись от клиента.
- Клиент. Задача обратится в салон за выполнением конкретной услуги.

## 2.6 Схема объектов базы данных

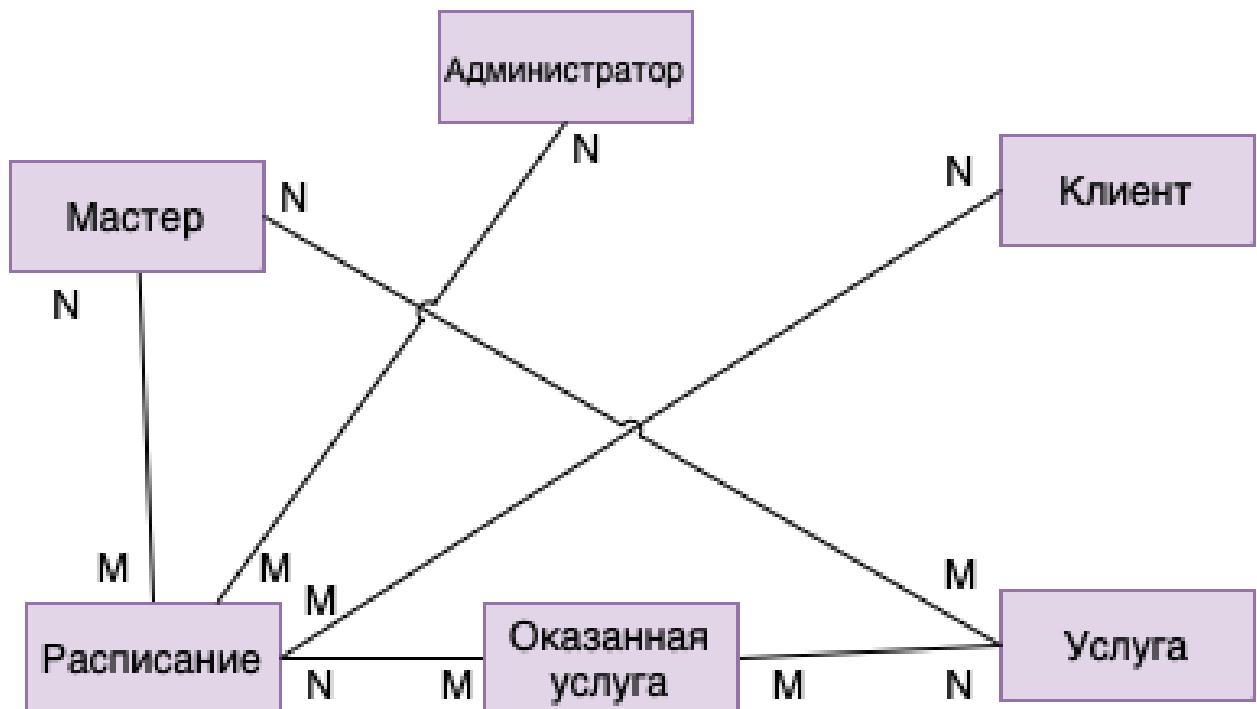


Рис. 2: Схема объектов базы данных

# 3 Проектирование

## 3.1 Лингвистическое описание

- Фамилия(<Фамилия\_id, INT>, <Фамилия, VARCHAR(45)>)
- Имя(<Имя\_id, INT>, <Имя, VARCHAR(45)>)
- Отчество(<Отчество\_id, INT>, <Отчество, VARCHAR(45)>)
- Мастер(<Мастер\_id, INT>, <Фамилия\_id, INT>, <Имя\_id, INT>, <Отчество\_id, INT>, <Мастер\_Опыт\_Работы, INT>, <Мастер\_Возраст, INT>)
- Клиент(<Клиент\_id, INT>, <Фамилия\_id, INT>, <Имя\_id, INT>, <Отчество\_id, INT>)
- Администратор(<Администратор\_id, INT>, <Фамилия\_id, INT>, <Имя\_id, INT>, <Отчество\_id, INT>)
- Услуга(<Услуга\_id, INT>, <Услуга\_Цена, INT>, <Услуга\_Название, VARCHAR(100)>, <Услуга\_Длительность, TIME>)
- Расписание(<Расписание\_id, INT>, <Расписание\_Дата, DATE>, <Время\_id, INT>, <Расписание\_Свободно, BOOLEAN>, <Мастер\_id, INT>, <Администратор\_id, INT>, <Клиент\_id, INT>)
- Время(<Время\_id, INT>, <Время, TEXT>)
- Оказанная услуга(<Оказанная\_Услуга\_id, INT>, <Услуга\_id, INT>, <Расписание\_id, INT>)
- Возможная услуга(<Возможная\_услуга\_id, INT>, <Мастер\_id, INT>, <Услуга\_id, INT>)

## 3.2 Таблицы метаданных

Персональные ключи имеют тип INT, чтобы с добавлением записи в базу данных идентификаторы автоматически увеличивались. Фамилии, имена и отчества хранятся в строке длины до 45 символов, поскольку наибольшая длина фамилии и отчества почти совпадает и равна 43-46 символов. Так как у человека может не быть отчества, то значение этого атрибута может быть не определено.

|    | Фамилия     | Surname     |              |          |
|----|-------------|-------------|--------------|----------|
|    | Название_RU | Название_EN | Тип атрибута | Значение |
| PK | Фамилия_id  | Surname_id  | INT          | NOT NULL |
|    | Фамилия     | Surname     | VARCHAR(45)  | NOT NULL |

Рис. 3: Таблица «Фамилия»

|    | Имя         | Name        |              |          |
|----|-------------|-------------|--------------|----------|
|    | Название_RU | Название_EN | Тип атрибута | Значение |
| PK | Имя_id      | Name_id     | INT          | NOT NULL |
|    | Имя         | Name        | VARCHAR(45)  | NOT NULL |

Рис. 4: Таблица «Имя»

|    | Отчество    | Patronymic    |              |          |
|----|-------------|---------------|--------------|----------|
|    | Название_RU | Название_EN   | Тип атрибута | Значение |
| PK | Отчество_id | Patronymic_id | INT          | NOT NULL |
|    | Отчество    | Patronymic    | VARCHAR(45)  | NULL     |

Рис. 5: Таблица «Отчество»

| Мастер |                    | Manicurist            |              |          |                            |
|--------|--------------------|-----------------------|--------------|----------|----------------------------|
|        | Название_RU        | Название_EN           | Тип атрибута | Значение |                            |
| PK     | Мастер_id          | Manicurist_id         | INT          | NOT NULL |                            |
|        | Фамилия_id         | Surname_id            | INT          | NOT NULL | Внешний ключ - Фамилия id  |
|        | Имя_id             | Name_id               | INT          | NOT NULL | Внешний ключ - Имя id      |
|        | Отчество_id        | Patronymic_id         | INT          | NOT NULL | Внешний ключ - Отчество id |
|        | Мастер_Опыт_Работы | Manicurist_Experience | INT          | NOT NULL |                            |
|        | Мастер_Возраст     | Manicurist_Age        | INT          | NOT NULL |                            |

Рис. 6: Таблица «Мастер»

| Клиент |             | Client        |              |          |                            |
|--------|-------------|---------------|--------------|----------|----------------------------|
|        | Название_RU | Название_EN   | Тип атрибута | Значение |                            |
| PK     | Клиент_id   | Client_id     | INT          | NOT NULL |                            |
|        | Фамилия_id  | Surname_id    | INT          | NOT NULL | Внешний ключ - Фамилия id  |
|        | Имя_id      | Name_id       | INT          | NOT NULL | Внешний ключ - Имя id      |
|        | Отчество_id | Patronymic_id | INT          | NOT NULL | Внешний ключ - Отчество id |

Рис. 7: Таблица «Клиент»

| Администратор |                  | Administrator    |              |          |                            |
|---------------|------------------|------------------|--------------|----------|----------------------------|
|               | Название_RU      | Название_EN      | Тип атрибута | Значение |                            |
| PK            | Администратор_id | Administrator_id | INT          | NOT NULL |                            |
|               | Фамилия_id       | Surname_id       | INT          | NOT NULL | Внешний ключ - Фамилия id  |
|               | Имя_id           | Name_id          | INT          | NOT NULL | Внешний ключ - Имя id      |
|               | Отчество_id      | Patronymic_id    | INT          | NOT NULL | Внешний ключ - Отчество id |

Рис. 8: Таблица «Администратор»

| Услуга |                     | Service          |              |          |
|--------|---------------------|------------------|--------------|----------|
|        | Название_RU         | Название_EN      | Тип атрибута | Значение |
| PK     | Услуга_id           | Service_id       | INT          | NOT NULL |
|        | Услуга_Цена         | Service_Price    | INT          | NOT NULL |
|        | Услуга_Название     | Service_Name     | VARCHAR(100) | NOT NULL |
|        | Услуга_Длительность | Service_Duration | TIME         | NOT NULL |

Рис. 9: Таблица «Услуга»

| Расписание |                     | Schedule         |              |          |                                 |  |
|------------|---------------------|------------------|--------------|----------|---------------------------------|--|
|            | Название_RU         | Название_EN      | Тип атрибута | Значение |                                 |  |
| PK         | Расписание_id       | Schedule_id      | INT          | NOT NULL |                                 |  |
|            | Расписание_Дата     | Schedule_Date    | DATE         | NOT NULL |                                 |  |
|            | Время_id            | Time_id          | INT          | NOT NULL |                                 |  |
|            | Расписание_Свободно | Schedule_Free    | BOOLEAN      | NOT NULL |                                 |  |
|            | Мастер_id           | Manicurist_id    | INT          | NOT NULL | Внешний ключ - Мастер id        |  |
|            | Администратор_id    | Administrator_id | INT          | NOT NULL | Внешний ключ - Администратор id |  |
|            | Клиент_id           | Client_id        | INT          | NOT NULL | Внешний ключ - Клиент id        |  |

Рис. 10: Таблица «Расписание»

| Время |             | Time        |              |          |  |
|-------|-------------|-------------|--------------|----------|--|
|       | Название_RU | Название_EN | Тип атрибута | Значение |  |
| PK    | Время_id    | Time_id     | INT          | NOT NULL |  |
|       | Время       | Time        | TEXT         | NOT NULL |  |

Рис. 11: Таблица «Время»

| Оukanная услуга |                     | Service provided    |              |                                       |
|-----------------|---------------------|---------------------|--------------|---------------------------------------|
|                 | Название_RU         | Название_EN         | Тип атрибута | Значение                              |
| PK              | Оказанная_Услуга_id | Service_provided_id | INT          | NOT NULL                              |
|                 | Услуга_id           | Service_id          | INT          | NOT NULL Внешний ключ - Услуга id     |
|                 | Расписание_id       | Schedule_id         | INT          | NOT NULL Внешний ключ - Расписание id |

Рис. 12: Таблица «Оказанная услуга»

| Возможная услуга |                     | Possible service    |              |                                   |
|------------------|---------------------|---------------------|--------------|-----------------------------------|
|                  | Название_RU         | Название_EN         | Тип атрибута | Значение                          |
| PK               | Возможная услуга_id | Possible service_id | INT          | NOT NULL                          |
|                  | Мастер_id           | Manicurist_id       | INT          | NOT NULL Внешний ключ - Мастер id |
|                  | Услуга_id           | Service_id          | INT          | NOT NULL Внешний ключ - Услуга id |

Рис. 13: Таблица «Возможная услуга»

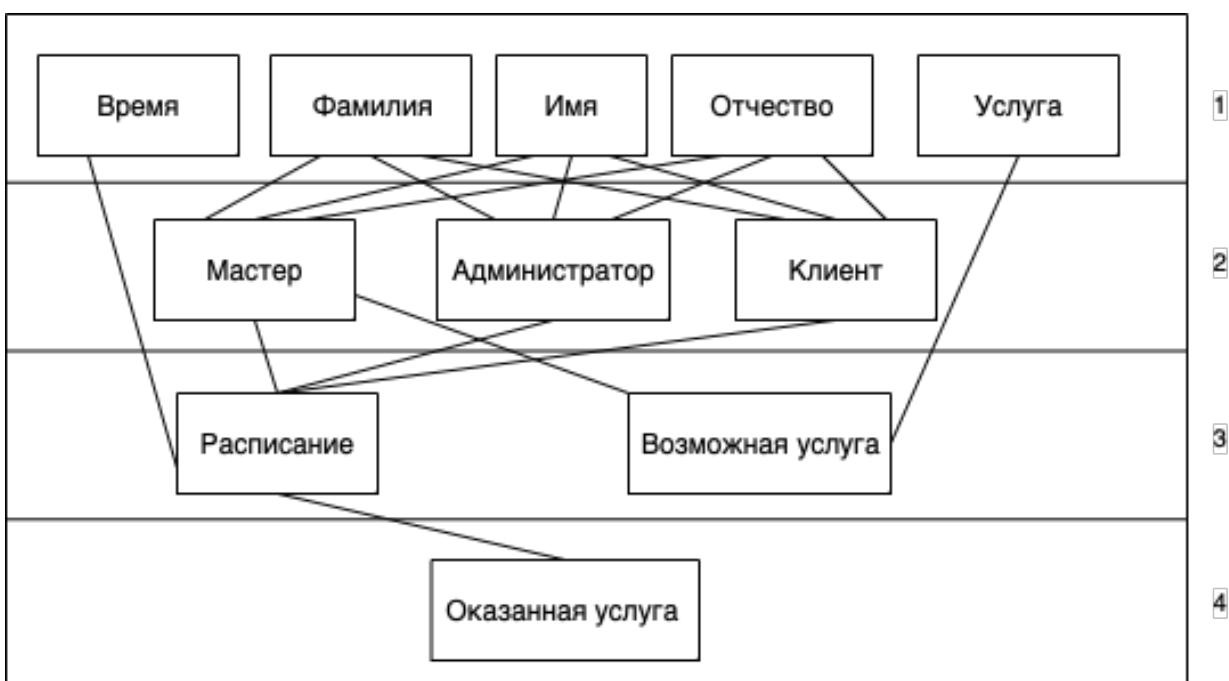


Рис. 14: Уровни заполнения базы данных

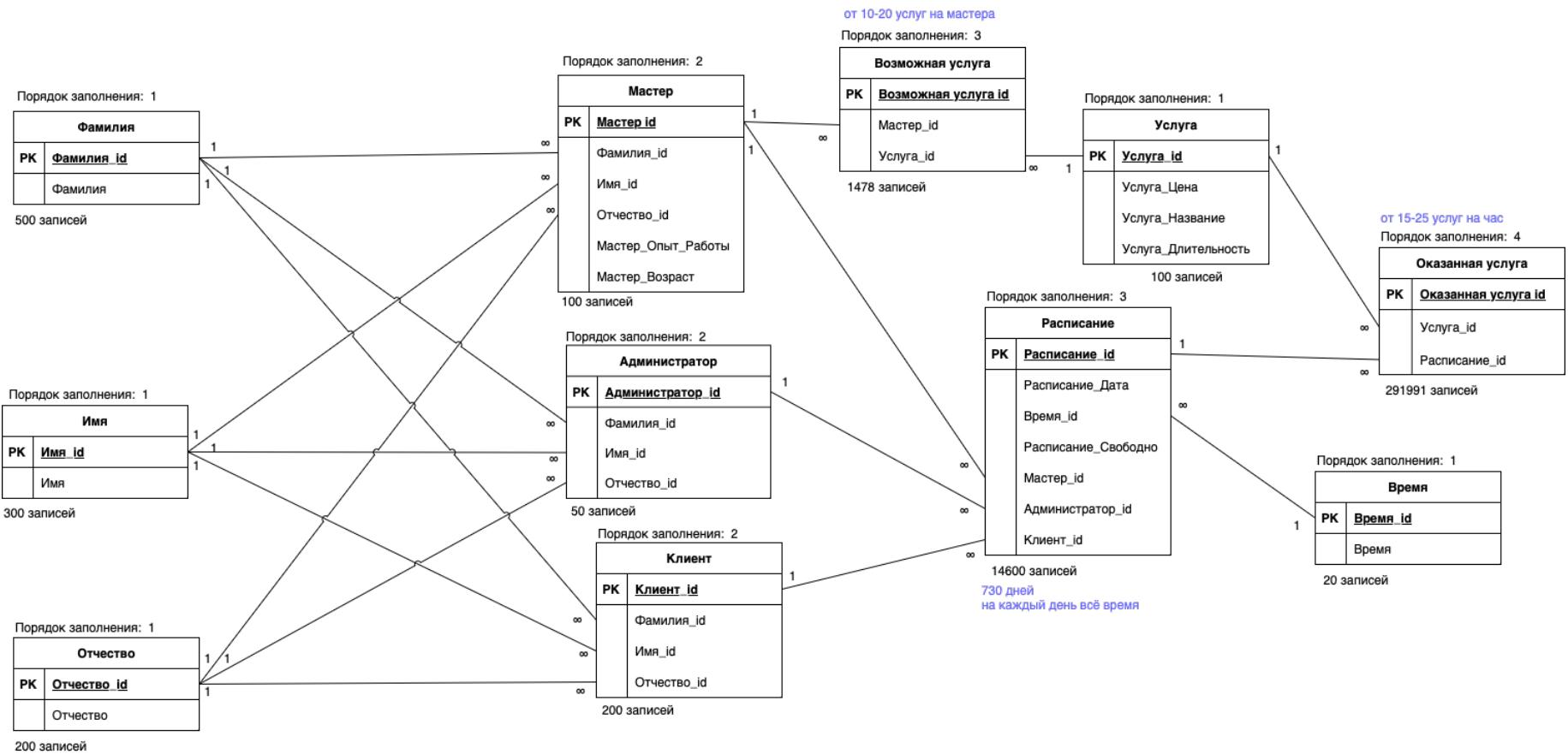


Рис. 15: Диаграмма базы данных на русском языке

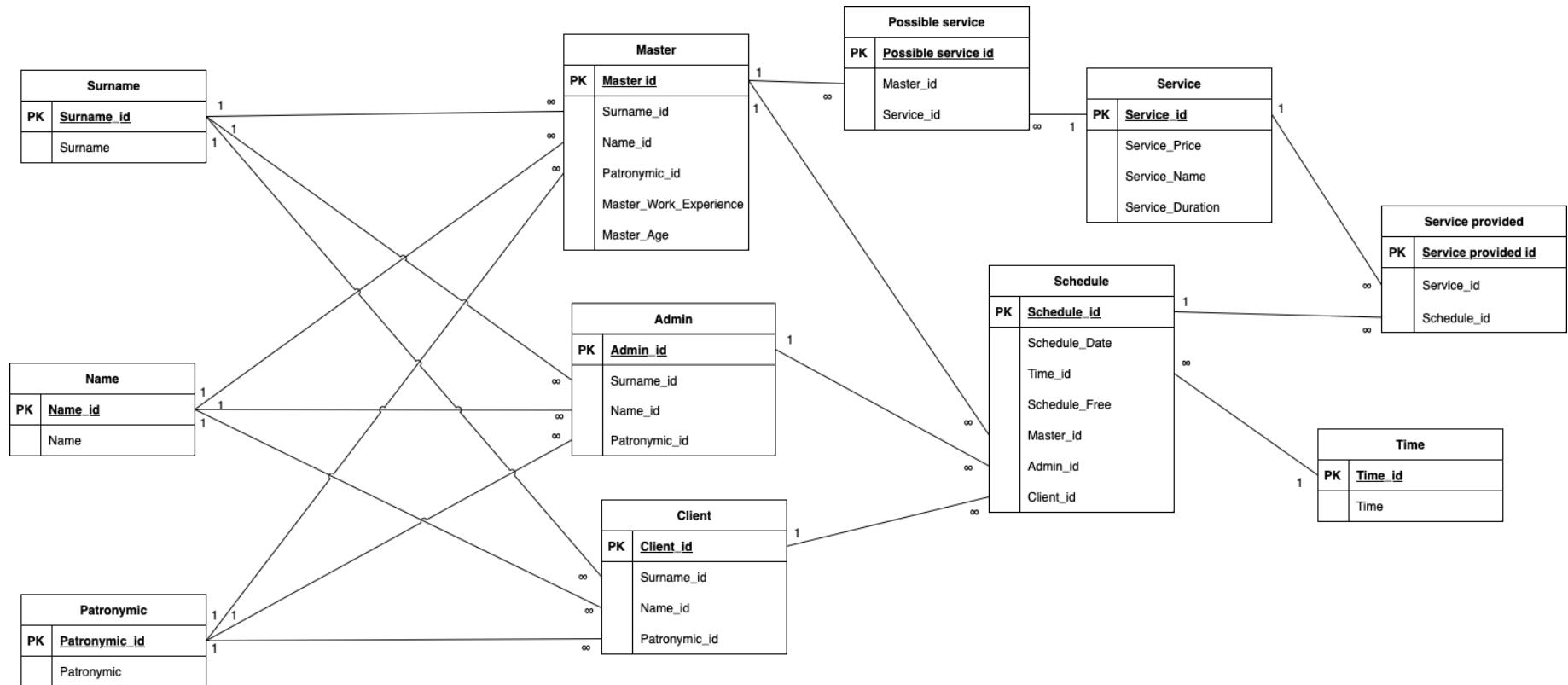


Рис. 16: Диаграмма базы данных на английском языке

### **3.3 Генерация записей**

Для генерации записей в базе данных был написан код на языке Python, который наполняет текстовый файл командами вставки INSERT. Также в начале создается соединение с базой данных. Данные заполняются циклами `for i in range(1, n)`. Таблицы типа «клиент» заполняются рандомным выбором id фамилии, имени и отчества с помощью `random.choice(name)`. Опыт и возраст командой `random.randint(1, n)`. `cursor.execute("SELECT COUNT(*) FROM client")` используется для подсчёта количества записей в таблице.

Код заполнения таблиц приведён в приложении В(4.8).

# 4 Программирование

## Лист проверки выполнения запросов курсовой работы по дисциплине «Теоретические основы баз данных» 4 семестр

Имя, отчество, фамилия Михайлова Анастасия Александровна Группа 5130201/3000\_2

Тема работы Самостоятельная работа

| №<br>п.п. | текст запроса  | принято  |                          |
|-----------|--|--|--------------------------|
|           |  | запрос   | отчет                    |
| 1         | Найти количество ком. получивших училишь членов по различным А с организаций училишь в ее време с С до В не менее 5 раз          | <input checked="" type="checkbox"/><br><input checked="" type="checkbox"/> | <input type="checkbox"/> |
| 1.2       |  |  |                          |
| 2         | Дни училишь А посещавшие наи-бо членов, неоднократно её посещали   | <input checked="" type="checkbox"/>  | <input type="checkbox"/> |
| 3         | Дни всех времена посещавшие сего членов, ког. однократно в это време, и, минимум, пять занес. любых где тоо време 2D, посещавшие | <input checked="" type="checkbox"/>  | <input type="checkbox"/> |
| 4         | Посещавшие гимн. членов с однократной минимум безминимум училишь. 2D посещавшие  | <input checked="" type="checkbox"/>  | <input type="checkbox"/> |
| 5         | Найти членов, ком. оказ. член. число училишь по зданию   | <input checked="" type="checkbox"/><br><input checked="" type="checkbox"/> | <input type="checkbox"/> |
| 5.1       | минимумное   |  |                          |
| 6         | Училишь аудит., ком. зарег. больше членов, чем аудитор А   | <input checked="" type="checkbox"/>  | <input type="checkbox"/> |
| 7         | Найти училишь, ком. число все оз. членов   | <input checked="" type="checkbox"/>  | <input type="checkbox"/> |
| 8         | Дни всех времена и первых 10 аудиторов посещавшие гимн. членов. 3D посещавшие  | <input checked="" type="checkbox"/>  | <input type="checkbox"/> |
| 9         |  | <input type="checkbox"/>   | <input type="checkbox"/> |
| 10        |  | <input type="checkbox"/>   | <input type="checkbox"/> |

## 4.1 Запрос 1

### 4.1.1 Вариант 1

Найти клиентов, которые получали услуги мастера по фамилии А, с оказанием услуги В, во время с С до D.

```
1      SELECT
2          c.client_id,
3          m.master_id,
4          m.surname_id AS master_surname_id,
5          ou.service_id AS service_id,
6          t.time
7      FROM
8          master m
9      JOIN
10         schedule sch ON m.master_id = sch.master_id
11     JOIN
12        client c ON sch.client_id = c.client_id
13     JOIN
14        okasusluga ou ON sch.schedule_id = ou.schedule_id
15     JOIN
16        service s ON ou.service_id = s.service_id
17     JOIN
18        time t ON sch.time_id = t.time_id
19     WHERE
20        m.master_id = 1
21     AND ou.service_id = 2
22     AND t.time_id BETWEEN 1 AND 5;
```

| client_id | master_id | master_surname_id | service_id | time        |
|-----------|-----------|-------------------|------------|-------------|
| 119       | 1         | 283               | 2          | 6:00 - 7:00 |
| 152       | 1         | 283               | 2          | 8:00 - 9:00 |
| 37        | 1         | 283               | 2          | 5:00 - 6:00 |
| 55        | 1         | 283               | 2          | 4:00 - 5:00 |
| 77        | 1         | 283               | 2          | 8:00 - 9:00 |
| 121       | 1         | 283               | 2          | 5:00 - 6:00 |
| 180       | 1         | 283               | 2          | 6:00 - 7:00 |
| 184       | 1         | 283               | 2          | 6:00 - 7:00 |
| 6         | 1         | 283               | 2          | 7:00 - 8:00 |
| 193       | 1         | 283               | 2          | 4:00 - 5:00 |
| 150       | 1         | 283               | 2          | 7:00 - 8:00 |
| 15        | 1         | 283               | 2          | 7:00 - 8:00 |

Рис. 17: Результаты запроса 1

Время выполнения запроса: 0:00:0.00491381.

| id | select_ty... | table | partitions | type   | possible_keys                       | key         | key_len | ref                    | rows | filtered | Extra       |
|----|--------------|-------|------------|--------|-------------------------------------|-------------|---------|------------------------|------|----------|-------------|
| 1  | SIMPLE       | m     | NULL       | const  | PRIMARY                             | PRIMARY     | 4       | const                  | 1    | 100.00   | NULL        |
| 1  | SIMPLE       | s     | NULL       | const  | PRIMARY                             | PRIMARY     | 4       | const                  | 1    | 100.00   | Using index |
| 1  | SIMPLE       | sch   | NULL       | ref    | PRIMARY,master_id,client_id,time_id | master_id   | 4       | const                  | 164  | 24.97    | Using where |
| 1  | SIMPLE       | ou    | NULL       | ref    | service_id,schedule_id              | schedule_id | 4       | mybase.sch.schedule_id | 20   | 1.00     | Using where |
| 1  | SIMPLE       | t     | NULL       | eq_ref | PRIMARY                             | PRIMARY     | 4       | mybase.sch.time_id     | 1    | 100.00   | NULL        |
| 1  | SIMPLE       | c     | NULL       | eq_ref | PRIMARY                             | PRIMARY     | 4       | mybase.sch.client_id   | 1    | 100.00   | Using index |

Рис. 18: Explain запроса 1

Запрос начинает работу с поиском в таблице master единственной строки, соответствующей условию master\_id = 1, используя первичный ключ для доступа. Сразу же после этого система обращается к таблице schedule, где через индекс по полю master\_id находит все записи расписания для данного мастера. Для каждого найденного элемента расписания запрос переходит к таблице client, извлекая информацию о клиенте через первичный ключ client\_id. Далее выполняется соединение с таблицей okasusluga по полю schedule\_id, где дополнительно применяется фильтр service\_id = 2, что позволяет системе отбросить все ненужные записи об услугах на этом этапе. Затем, используя первичный ключ происходит обращение к таблице time, где выбираются только те временные интервалы, чьи идентификаторы попадают в указанный диапазон от 1 до 5.

#### 4.1.2 Вариант 2

Найти клиентов, которые получали услуги мастера по фамилии А, с оказанием услуги В, во время с С до D не менее 5 раз.

```

1      SELECT
2          c.client_id,
3              c.surname_id AS client_surname_id,
4              c.name_id AS client_name_id,
5              c.patronymic_id AS client_patronymic_id,
6          m.master_id,
7          ou.service_id AS service_id
8      FROM
9          master m
10     JOIN
11         schedule sch ON m.master_id = sch.master_id
12     JOIN
13         client c ON sch.client_id = c.client_id
14     JOIN
15         okasusluga ou ON sch.schedule_id = ou.schedule_id
16     JOIN
17         service s ON ou.service_id = s.service_id
18     JOIN
19         time t ON sch.time_id = t.time_id
20     WHERE
21         m.master_id = 7
22         AND ou.service_id = 35
23         AND t.time_id BETWEEN 1 AND 20

```

```

24     GROUP BY
25         c.client_id
26     HAVING
27         COUNT(sch.schedule_id) >= 5;

```

Время выполнения запроса: 0:00:0.01087403.

| id | select_ty... | table | partitions | type   | possible_keys                               | key         | key_len | ref                    |      |          |                              |
|----|--------------|-------|------------|--------|---|-------------|---------|------------------------|------|----------|------------------------------|
|    |              |       |            |        |   |             |         |                        | rows | filtered | Extra                        |
| 1  | SIMPLE       | m     | NULL       | const  | PRIMARY                                     | PRIMARY     | 4       | const                  | 1    | 100.00   | Using index; Using temporary |
| 1  | SIMPLE       | s     | NULL       | const  | PRIMARY                                     | PRIMARY     | 4       | const                  | 1    | 100.00   | Using index                  |
| 1  | SIMPLE       | sch   | NULL       | ref    | PRIMARY, master_id, client_id, time_id      | master_id   | 4       | const                  | 144  | 50.00    | Using where                  |
| 1  | SIMPLE       | ou    | NULL       | ref    | service_id, schedule_id                     | schedule_id | 4       | mybase.sch.schedule_id | 20   | 0.98     | Using where                  |
| 1  | SIMPLE       | t     | NULL       | eq_ref | PRIMARY                                     | PRIMARY     | 4       | mybase.sch.time_id     | 1    | 100.00   | Using index                  |
| 1  | SIMPLE       | c     | NULL       | eq_ref | PRIMARY, surname_id, name_id, patronymic_id | PRIMARY     | 4       | mybase.sch.client_id   | 1    | 100.00   | NULL                         |

Рис. 19: Explain запроса 1.2

В отличие от первого варианта запроса здесь группируются полученные данные по идентификаторам клиентов (client\_id) и применяется условие HAVING, которое оставляет в результатах только тех клиентов, у которых количество записей (COUNT(sch.schedule\_id)) составляет 5 и более.

## 4.2 Запрос 2

Для услуги А посчитать количество клиентов, которые её получили.

```

1      SELECT COUNT(s.client_id) AS client_count
2          FROM schedule s
3      JOIN okasusluga o ON s.schedule_id = o.schedule_id
4      WHERE o.service_id = 5;

```

client\_count

2920

Рис. 20: Результаты запроса 2

Время выполнения запроса: 0:00:0.04188704.

| id | select_ty... | table | partitions | type   | possible_keys           | key        | key_len | ref                  |      |          |       |
|----|--------------|-------|------------|--------|-------------------------|------------|---------|----------------------|------|----------|-------|
|    |              |       |            |        |                         |            |         |                      | rows | filtered | Extra |
| 1  | SIMPLE       | o     | NULL       | ref    | service_id, schedule_id | service_id | 4       | const                | 2920 | 100.00   | NULL  |
| 1  | SIMPLE       | s     | NULL       | eq_ref | PRIMARY                 | PRIMARY    | 4       | mybase.o.schedule_id | 1    | 100.00   | NULL  |

Рис. 21: Explain запроса 2

Этот запрос выполняет подсчет общего количества записей клиентов, воспользовавшихся услугой с id 5. Обработка начинается с таблицы «okasusluga», где сразу применяется фильтр по полю service\_id, отбирая только те записи, которые соответствуют требуемой услуге (service\_id = 5). Затем полученный промежуточный результат соединяется с таблицей "schedule" через общее поле schedule\_id. После формирования полного набора данных подсчитывается количество значений в поле client\_id из таблицы расписания.

Реляционная формула:

$$\pi_{COUNT(s.client\_id) \rightarrow client\_count}(\sigma_{o.service\_id=5}(schedule \bowtie_{s.schedule\_id=o.schedule\_id} okasusluga))$$

### 4.3 Запрос 3

Для всех времен посчитать число клиентов, которые обслуживались в это время и число администраторов, которые записали людей на это время.

```

1      SELECT
2          t.time AS appointment_time ,
3          (SELECT COUNT(DISTINCT s.client_id)
4              FROM schedule s
5              WHERE s.time_id = t.time_id) AS client_count ,
6          (SELECT COUNT(DISTINCT s.admin_id)
7              FROM schedule s
8              WHERE s.time_id = t.time_id) AS admin_count
9
10         FROM
11             time t;
```

| appointment_time | client_count | admin_count |
|------------------|--------------|-------------|
| 4:00 - 5:00      | 195          | 50          |
| 5:00 - 6:00      | 195          | 50          |
| 6:00 - 7:00      | 194          | 50          |
| 7:00 - 8:00      | 190          | 50          |
| 8:00 - 9:00      | 195          | 50          |
| 9:00 - 10:00     | 196          | 50          |
| 10:00 - 11:00    | 193          | 50          |
| 11:00 - 12:00    | 194          | 50          |
| 12:00 - 13:00    | 191          | 50          |
| 13:00 - 14:00    | 191          | 50          |
| 14:00 - 15:00    | 196          | 50          |
| 15:00 - 16:00    | 191          | 50          |
| 16:00 - 17:00    | 193          | 50          |
| 17:00 - 18:00    | 194          | 50          |
| 18:00 - 19:00    | 196          | 50          |
| 19:00 - 20:00    | 196          | 50          |
| 20:00 - 21:00    | 195          | 50          |
| 21:00 - 22:00    | 191          | 50          |
| 22:00 - 23:00    | 193          | 50          |
| 23:00 - 24:00    | 197          | 50          |

Рис. 22: Результаты запроса 3

Время выполнения запроса: 0:00:0.04979300.

| id | select_ty... | table | partitions | type | possible_keys     | key     | key_len | ref              | rows | filtered | Extra |
|----|--------------|-------|------------|------|-------------------|---------|---------|------------------|------|----------|-------|
| 1  | PRIMARY      | t     | NULL       | ALL  | NULL              | NULL    | NULL    | NULL             | 20   | 100.00   | NULL  |
| 3  | DEPEND...    | s     | NULL       | ref  | admin_id,time_id  | time_id | 4       | mybase.t.time_id | 730  | 100.00   | NULL  |
| 2  | DEPEND...    | s     | NULL       | ref  | client_id,time_id | time_id | 4       | mybase.t.time_id | 730  | 100.00   | NULL  |

Рис. 23: Explain запроса 3

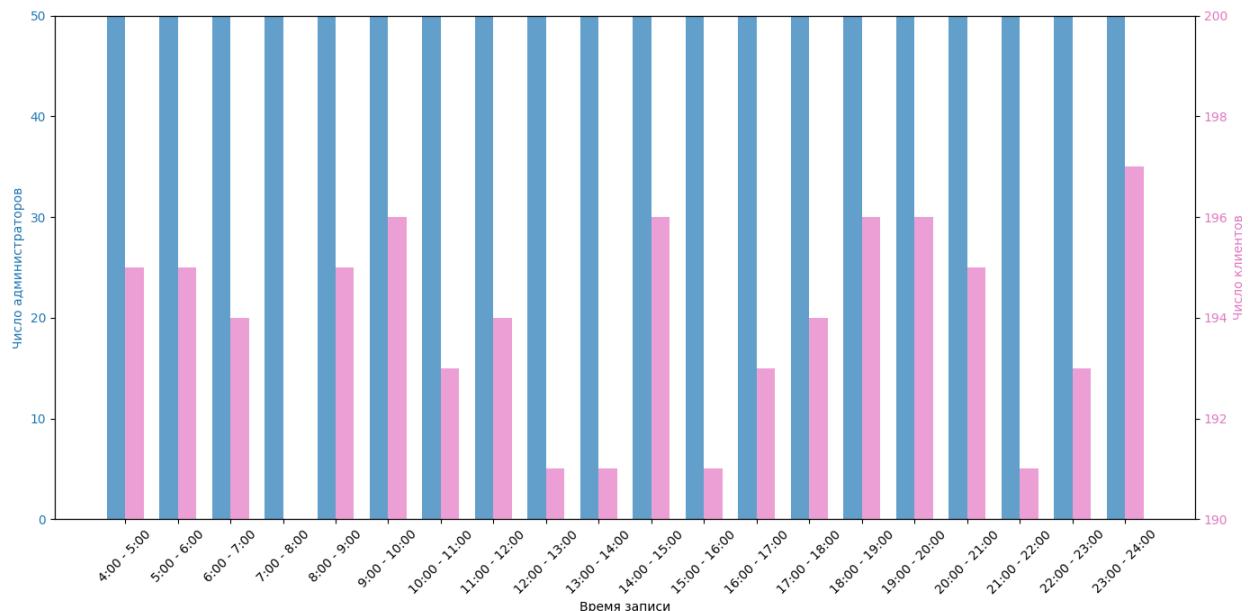


Рис. 24: Диаграмма результатов запроса 3

Данный запрос выполняет соединение таблицы time с таблицей schedule через поле time\_id, чтобы получить информацию о количестве клиентов и администраторов, запланированных на каждое время приема. Основная таблица time содержит список временных слотов, а подзапросы подсчитывают уникальные (DISTINCT) значения client\_id и admin\_id из таблицы schedule для каждого временного интервала, используя условие совпадения s.time\_id = t.time\_id.

```


$$\pi_{\text{appointment\_time}, \text{client\_count}, \text{admin\_count}}($$

    time  $\bowtie$ 
    (COUNT(DISTINCT  $\pi_{\text{client\_id}}(\sigma(s.\text{time\_id} = t.\text{time\_id})(\text{schedule}))$ ) AS client_count),
    (COUNT(DISTINCT  $\pi_{\text{admin\_id}}(\sigma(s.\text{time\_id} = t.\text{time\_id})(\text{schedule}))$ ) AS admin_count)
)

```

## 4.4 Запрос 4

Посчитать число мастеров с одинаковым числом возможных услуг.

```

1   SELECT service_count, COUNT(master_id) AS master_count
2   FROM (
3       SELECT m.master_id, COUNT(v.service_id) AS service_count
4       FROM master m

```

```

5     LEFT JOIN vozmusluga v ON m.master_id = v.master_id
6     GROUP BY m.master_id
7   ) AS master_services
8   GROUP BY service_count;

```

| service_count | master_count |
|---------------|--------------|
| 20            | 8            |
| 12            | 11           |
| 17            | 7            |
| 10            | 11           |
| 15            | 7            |
| 13            | 12           |
| 16            | 6            |
| 19            | 13           |
| 14            | 12           |
| 11            | 7            |
| 18            | 6            |

Рис. 25: Результаты запроса 4

Время выполнения запроса: 0:00:0.00347185.

| id | select_ty... | table      | partitions | type  | possible_keys                            | key       | key_len | ref                | rows | filtered | Extra           |
|----|--------------|------------|------------|-------|--|-----------|---------|--------------------|------|----------|-----------------|
| 1  | PRIMARY      | <derived2> | NULL       | ALL   | NULL                                     | NULL      | NULL    | NULL               | 1477 | 100.00   | Using temporary |
| 2  | DERIVED      | m          | NULL       | index | PRIMARY,surname_id,name_id,patronymic_id | PRIMARY   | 4       | NULL               | 100  | 100.00   | Using index     |
| 2  | DERIVED      | v          | NULL       | ref   | master_id                                | master_id | 4       | mybase.m.master_id | 14   | 100.00   | NULL            |

Рис. 26: Explain запроса 4

Внутренний подзапрос соединяет таблицу master с таблицей vozmusluga через поле master\_id с использованием LEFT JOIN, чтобы включить всех мастеров, даже если у них нет закрепленных услуг. Для каждого мастера подсчитывается количество связанных с ним услуг service\_count. Затем внешний запрос группирует результаты по количеству услуг service\_count и подсчитывает, сколько мастеров master\_count соответствует каждой группе.

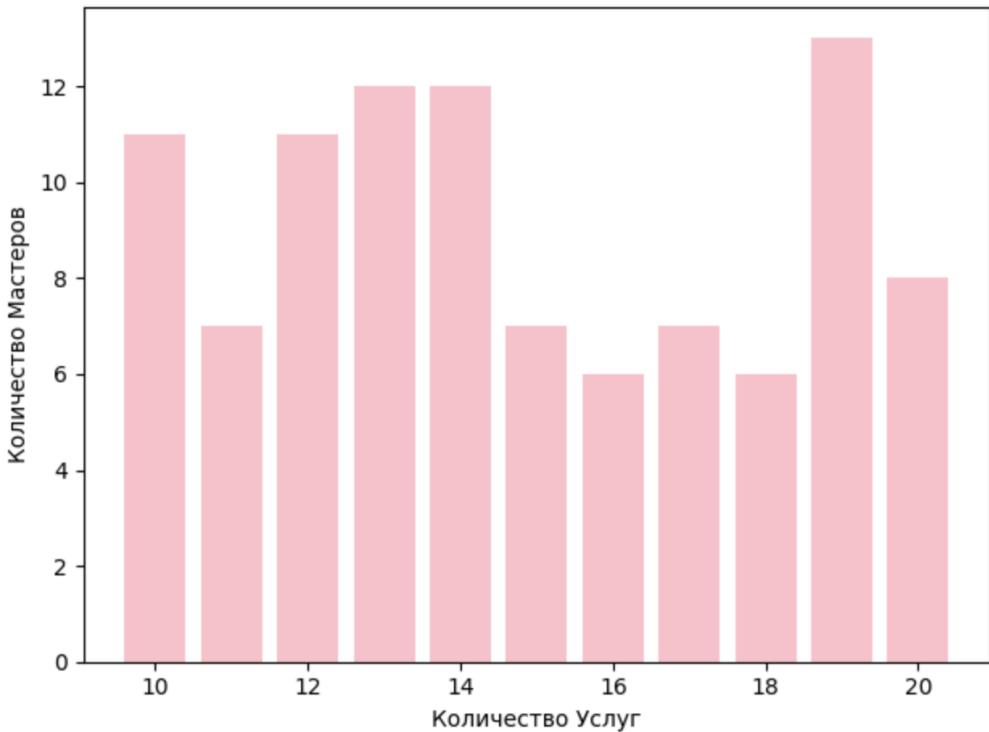


Рис. 27: Диаграмма результатов запроса 4

## 4.5 Запрос 5

### 4.5.1 Вариант 1

Найти мастеров, которые оказали максимальное число услуг.

```

1   SELECT
2       m.master_id,
3       CONCAT(surname.surname, ' ', n.name, ' ', p.patronymic)
4           AS master_full_name,
5       COUNT(o.okasusluga_id) AS service_count
6   FROM
7       master m
8   JOIN
9       vozmusluga v ON m.master_id = v.master_id
10  JOIN
11      okasusluga o ON v.service_id = o.service_id
12  JOIN
13      surname surname ON m.surname_id = surname.surname_id
14  JOIN
15      name n ON m.name_id = n.name_id
16  JOIN
17      patronymic p ON m.patronymic_id = p.patronymic_id
18  GROUP BY
19      m.master_id
20  HAVING
21      service_count = (

```

```

21     SELECT MAX(service_count)
22     FROM (
23         SELECT COUNT(o.okasusluga_id) AS service_count
24         FROM master m
25     JOIN vozmusluga v ON m.master_id = v.master_id
26     JOIN okasusluga o ON v.service_id = o.service_id
27     GROUP BY m.master_id
28 ) AS subquery
29 );

```

| master_id | master_full_name                | service_count |
|-----------|---------------------------------|---------------|
| 76        | Фамилия 362 Имя 186 Отчество 27 | 58580         |

Рис. 28: Результаты запроса 5

Время выполнения запроса: 0:00:0.79697990.

| id | select_ty... | table      | partitions | type   | possible_keys                            | key        | key_len | ref                    | rows    | filtered | Extra       |
|----|--------------|------------|------------|--------|--|------------|---------|------------------------|---------|----------|-------------|
| 1  | PRIMARY      | m          | NULL       | index  | PRIMARY,surname_id,name_id,patronymic_id | PRIMARY    | 4       | NULL                   | 100     | 100.00   | NULL        |
| 1  | PRIMARY      | p          | NULL       | eq_ref | PRIMARY                                  | PRIMARY    | 4       | mybase.m.patronymic_id | 1       | 100.00   | NULL        |
| 1  | PRIMARY      | n          | NULL       | eq_ref | PRIMARY                                  | PRIMARY    | 4       | mybase.m.name_id       | 1       | 100.00   | NULL        |
| 1  | PRIMARY      | surname    | NULL       | eq_ref | PRIMARY                                  | PRIMARY    | 4       | mybase.m.surname_id    | 1       | 100.00   | NULL        |
| 1  | PRIMARY      | v          | NULL       | ref    | master_id,service_id                     | master_id  | 4       | mybase.m.master_id     | 14      | 100.00   | NULL        |
| 1  | PRIMARY      | o          | NULL       | ref    | service_id                               | service_id | 4       | mybase.v.service_id    | 3071    | 100.00   | Using index |
| 2  | SUBQUE...    | <derived3> | NULL       | ALL    | NULL                                     | NULL       | NULL    |                        | 4540229 | 100.00   | NULL        |
| 3  | DERIVED      | m          | NULL       | index  | PRIMARY,surname_id,name_id,patronymic_id | PRIMARY    | 4       | NULL                   | 100     | 100.00   | Using index |
| 3  | DERIVED      | v          | NULL       | ref    | master_id,service_id                     | master_id  | 4       | mybase.m.master_id     | 14      | 100.00   | NULL        |
| 3  | DERIVED      | o          | NULL       | ref    | service_id                               | service_id | 4       | mybase.v.service_id    | 3071    | 100.00   | Using index |

Рис. 29: Explain запроса 5

Соединяется таблица master с таблицами vozmusluga и okasusluga через master\_id и service\_id, а также объединяется с таблицами surname, name и patronymic для формирования полного имени мастера. Затем группируются данные по master\_id и подсчитывается количество оказанных услуг (okasusluga\_id) для каждого мастера. В подзапросе вычисляется максимальное количество услуг среди всех мастеров, после чего в основном запросе с помощью HAVING выбираются только те мастера, количество услуг которых равно этому максимальному значению.

#### 4.5.2 Вариант 2

Найти мастеров, которые оказали минимальное число услуг.

```

1      SELECT
2          m.master_id,
3          CONCAT(surname.surname, ' ', n.name, ' ', p.patronymic)
4              AS master_full_name,
5          COUNT(o.okasusluga_id) AS service_count
6      FROM
7          master m
8      JOIN
9

```

```

8      vozmusluga v ON m.master_id = v.master_id
9      JOIN
10     okasusluga o ON v.service_id = o.service_id
11     JOIN
12     surname surname ON m.surname_id = surname.surname_id
13     JOIN
14     name n ON m.name_id = n.name_id
15     JOIN
16     patronymic p ON m.patronymic_id = p.patronymic_id
17     GROUP BY
18     m.master_id
19     HAVING
20     service_count = (
21     SELECT MIN(service_count)
22     FROM (
23     SELECT COUNT(o.okasusluga_id) AS service_count
24     FROM master m
25     JOIN vozmusluga v ON m.master_id = v.master_id
26     JOIN okasusluga o ON v.service_id = o.service_id
27     GROUP BY m.master_id
28   ) AS subquery
29 );

```

| master_id | master_full_name                 | service_count |
|-----------|----------------------------------|---------------|
| 46        | Фамилия 345 Имя 205 Отчество 198 | 28887         |

Рис. 30: Результаты запроса 5.1

Время выполнения запроса: 0:00:0.81372809.

| id | select_ty... | table      | partitions | type   | possible_keys                            | key        | key_len | ref                    | rows    | filtered | Extra       |
|----|--------------|------------|------------|--------|--|------------|---------|------------------------|---------|----------|-------------|
| 1  | PRIMARY      | m          | NULL       | index  | PRIMARY,surname_id,name_id,patronymic_id | PRIMARY    | 4       | NULL                   | 100     | 100.00   | HULL        |
| 1  | PRIMARY      | p          | NULL       | eq_ref | PRIMARY                                  | PRIMARY    | 4       | mybase.m.patronymic_id | 1       | 100.00   | HULL        |
| 1  | PRIMARY      | n          | NULL       | eq_ref | PRIMARY                                  | PRIMARY    | 4       | mybase.m.name_id       | 1       | 100.00   | HULL        |
| 1  | PRIMARY      | surname    | NULL       | eq_ref | PRIMARY                                  | PRIMARY    | 4       | mybase.m.surname_id    | 1       | 100.00   | HULL        |
| 1  | PRIMARY      | v          | NULL       | ref    | master_id,service_id                     | master_id  | 4       | mybase.m.master_id     | 14      | 100.00   | HULL        |
| 1  | PRIMARY      | o          | NULL       | ref    | service_id                               | service_id | 4       | mybase.v.service_id    | 3071    | 100.00   | Using index |
| 2  | SUBQUE...    | <derived3> | NULL       | ALL    | NULL                                     | NULL       | NULL    | NULL                   | 4540229 | 100.00   | HULL        |
| 3  | DERIVED      | m          | NULL       | index  | PRIMARY,surname_id,name_id,patronymic_id | PRIMARY    | 4       | NULL                   | 100     | 100.00   | Using index |
| 3  | DERIVED      | v          | NULL       | ref    | master_id,service_id                     | master_id  | 4       | mybase.m.master_id     | 14      | 100.00   | HULL        |
| 3  | DERIVED      | o          | NULL       | ref    | service_id                               | service_id | 4       | mybase.v.service_id    | 3071    | 100.00   | Using index |

Рис. 31: Explain запроса 5.1

Соединяется таблица master с таблицами vozmusluga и okasusluga через master\_id и service\_id, а также объединяется с таблицами surname, name и patronymic для формирования полного имени мастера. Затем группируются данные по master\_id и подсчитывается количество оказанных услуг (okasusluga\_id) для каждого мастера. В подзапросе вычисляется минимальное количество услуг среди всех мастеров, после чего в основном запросе с помощью HAVING выбираются только те мастера, количество услуг которых равно этому минимальному значению.

## 4.6 Запрос 6

Найти администраторов, которые зарегистрировали больше клиентов, чем администратор А.

```
1   SELECT
2     a.admin_id,
3       CONCAT(surname.surname, ' ', name.name, ' ', patronymic.
4           patronymic) AS admin_full_name,
5           COUNT(c.client_id) AS client_count
6   FROM
7     admin a
8   JOIN
9     schedule s ON a.admin_id = s.admin_id
10  JOIN
11    client c ON s.client_id = c.client_id
12  JOIN
13    surname ON a.surname_id = surname.surname_id
14  JOIN
15    name ON a.name_id = name.name_id
16  JOIN
17    patronymic ON a.patronymic_id = patronymic.patronymic_id
18  GROUP BY
19    a.admin_id
20  HAVING
21    client_count > (
22      SELECT COUNT(c2.client_id)
23      FROM schedule s2
24      JOIN client c2 ON s2.client_id = c2.client_id
25      WHERE s2.admin_id = 1
26    );
```

| admin_id | admin_full_name                  | client_count |
|----------|----------------------------------|--------------|
| 4        | Фамилия 454 Имя 22 Отчество 162  | 325          |
| 5        | Фамилия 158 Имя 205 Отчество 137 | 298          |
| 6        | Фамилия 393 Имя 190 Отчество 92  | 316          |
| 11       | Фамилия 356 Имя 187 Отчество 11  | 300          |
| 13       | Фамилия 161 Имя 16 Отчество 70   | 294          |
| 14       | Фамилия 374 Имя 224 Отчество 27  | 299          |
| 15       | Фамилия 167 Имя 260 Отчество 29  | 307          |
| 16       | Фамилия 356 Имя 101 Отчество 157 | 303          |
| 17       | Фамилия 65 Имя 75 Отчество 36    | 327          |
| 19       | Фамилия 370 Имя 24 Отчество 119  | 294          |
| 21       | Фамилия 442 Имя 187 Отчество 87  | 312          |
| 22       | Фамилия 257 Имя 106 Отчество 87  | 305          |
| 24       | Фамилия 498 Имя 122 Отчество 77  | 329          |
| 28       | Фамилия 12 Имя 92 Отчество 57    | 293          |
| 31       | Фамилия 83 Имя 192 Отчество 158  | 314          |
| 32       | Фамилия 336 Имя 153 Отчество 140 | 298          |
| 34       | Фамилия 83 Имя 88 Отчество 94    | 309          |
| 35       | Фамилия 337 Имя 18 Отчество 83   | 298          |
| 36       | Фамилия 70 Имя 237 Отчество 23   | 293          |
| 39       | Фамилия 248 Имя 106 Отчество 65  | 319          |
| 40       | Фамилия 425 Имя 159 Отчество 29  | 313          |
| 43       | Фамилия 10 Имя 291 Отчество 91   | 307          |
| 44       | Фамилия 394 Имя 139 Отчество 180 | 299          |
| 45       | Фамилия 69 Имя 240 Отчество 138  | 314          |
| 47       | Фамилия 216 Имя 82 Отчество 165  | 303          |

Рис. 32: Результаты запроса 6

Время выполнения запроса: 0:00:0.04077721.

| id | select_ty... | table      | partitions | type   | possible_keys                            | key      | key_len | ref                    | rows | filtered | Extra       |
|----|--------------|------------|------------|--------|--|----------|---------|------------------------|------|----------|-------------|
| 1  | PRIMARY      | a          | NULL       | index  | PRIMARY,surname_id,name_id,patronymic_id | PRIMARY  | 4       | NULL                   | 50   | 100.00   | NULL        |
| 1  | PRIMARY      | patronymic | NULL       | eq_ref | PRIMARY                                  | PRIMARY  | 4       | mybase.a.patronymic_id | 1    | 100.00   | NULL        |
| 1  | PRIMARY      | name       | NULL       | eq_ref | PRIMARY                                  | PRIMARY  | 4       | mybase.a.name_id       | 1    | 100.00   | NULL        |
| 1  | PRIMARY      | surname    | NULL       | eq_ref | PRIMARY                                  | PRIMARY  | 4       | mybase.a.surname_id    | 1    | 100.00   | NULL        |
| 1  | PRIMARY      | s          | NULL       | ref    | admin_id,client_id                       | admin_id | 4       | mybase.a.admin_id      | 292  | 100.00   | NULL        |
| 1  | PRIMARY      | c          | NULL       | eq_ref | PRIMARY                                  | PRIMARY  | 4       | mybase.s.client_id     | 1    | 100.00   | Using index |
| 2  | SUBQUE...    | s2         | NULL       | ref    | admin_id,client_id                       | admin_id | 4       | const                  | 292  | 100.00   | NULL        |
| 2  | SUBQUE...    | c2         | NULL       | eq_ref | PRIMARY                                  | PRIMARY  | 4       | mybase.s2.client_id    | 1    | 100.00   | Using index |

Рис. 33: Explain запроса 6

Соединяется таблица admin с таблицей schedule по полю admin\_id, чтобы получить все записи расписания, связанные с каждым администратором. Далее запрос связывает таблицу schedule с таблицей client через поле client\_id, устанавливая соответствие между администраторами и их клиентами. Для формирования полного имени администратора дополнительно подключаются таблицы: surname, name и patronymic. Функция COUNT(c.client\_id) подсчитывает общее количество клиентов у каждого администратора. HAVING сравнивает полученное количество клиентов с результатом подзапроса. Этот подзапрос выполняет аналогичные соединения таблиц, но считает клиентов только для администратора с admin\_id = 1.

## 4.7 Запрос 7

Найти услуги, которые никогда не оказывались клиентом.

```

1   SELECT
2       s.service_id ,
3       s.service_name
4   FROM
5       service s
6   LEFT JOIN
7       okasusluga o ON s.service_id = o.service_id
8   WHERE
9       o.okasusluga_id IS NULL ;

```

| service_id | service_name |
|------------|--------------|
| 101        | Услуга 101   |

Рис. 34: Результаты запроса 7

Время выполнения запроса: 0:00:0.01136088.

| id | select_ty... | table | partitions | type | possible_keys | key        | key_len | ref                 | rows | filtered | Extra                                |
|----|--------------|-------|------------|------|---------------|------------|---------|---------------------|------|----------|--------------------------------------|
| 1  | SIMPLE       | s     |            | ALL  | NULL          | NULL       | NULL    | NULL                | 100  | 100.00   | NULL                                 |
| 1  | SIMPLE       | o     |            | ref  | service_id    | service_id | 4       | mybase.s.service_id | 3071 | 10.00    | Using where; Not exists; Using index |

Рис. 35: Explain запроса 7

Запрос находит невостребованные услуги, выбирая service\_id и service\_name из таблицы service через LEFT JOIN с таблицей okasusluga по полю service\_id, где условие o.okasusluga\_id IS NULL отфильтровывает только те услуги, которые отсутствуют в таблице оказанных услуг.

Реляционная формула:

$$\pi_{\text{service\_id}, \text{service\_name}} \left( \sigma_{\text{okasusluga\_id} = \text{null}} \left( \text{service} \bowtie_{\text{service\_id} = \text{service\_id}} \text{okasusluga} \right) \right)$$

## 4.8 Запрос 8

Для всех времен и первых 10 админов посчитать число клиентов.

```

1      SELECT
2          a.admin_id,
3          t.time,
4          (SELECT COUNT(s.client_id)
5           FROM schedule s
6           WHERE s.admin_id = a.admin_id AND s.time_id = t.time_id)
7           AS client_count
8
9      FROM
10     admin a
11     CROSS JOIN
12     time t
13     WHERE
14     a.admin_id <= 10;

```

| admin_id | time        | client_count |
|----------|-------------|--------------|
| 10       | 4:00 - 5:00 | 16           |
| 9        | 4:00 - 5:00 | 9            |
| 8        | 4:00 - 5:00 | 12           |
| 7        | 4:00 - 5:00 | 15           |
| 6        | 4:00 - 5:00 | 13           |
| 5        | 4:00 - 5:00 | 13           |
| 4        | 4:00 - 5:00 | 11           |
| 3        | 4:00 - 5:00 | 14           |
| 2        | 4:00 - 5:00 | 15           |
| 1        | 4:00 - 5:00 | 18           |
| 10       | 5:00 - 6:00 | 8            |
| 9        | 5:00 - 6:00 | 19           |
| 8        | 5:00 - 6:00 | 13           |
| 7        | 5:00 - 6:00 | 18           |
| 6        | 5:00 - 6:00 | 19           |

Рис. 36: Фрагмент результатов запроса 8

Время выполнения запроса: 0:00:0.06207585.

| id | select_ty... | table | partitions | type  | possible_keys    | key      | key_len | ref               | rows | filtered | Extra                         |
|----|--------------|-------|------------|-------|------------------|----------|---------|-------------------|------|----------|-------------------------------|
| 1  | PRIMARY      | a     | HULL       | range | PRIMARY          | PRIMARY  | 4       | HULL              | 10   | 100.00   | Using where; Using index      |
| 1  | PRIMARY      | t     | HULL       | ALL   | NULL             | HULL     | HULL    | HULL              | 20   | 100.00   | Using join buffer (hash join) |
| 2  | DEPEND...    | s     | HULL       | ref   | admin_id,time_id | admin_id | 4       | mybase.a.admin_id | 292  | 10.00    | Using where                   |

Рис. 37: Explain запроса 8

Соединяются таблицы admin и time через CROSS JOIN, создавая все возможные комбинации администраторов и временных интервалов. Для каждой комбинации считается количество клиентов (client\_count) через подзапрос к таблице schedule, где проверяется соответствие по admin\_id и time\_id. Условие WHERE a.admin\_id <= 10 ограничивает выборку первыми 10 администраторами.

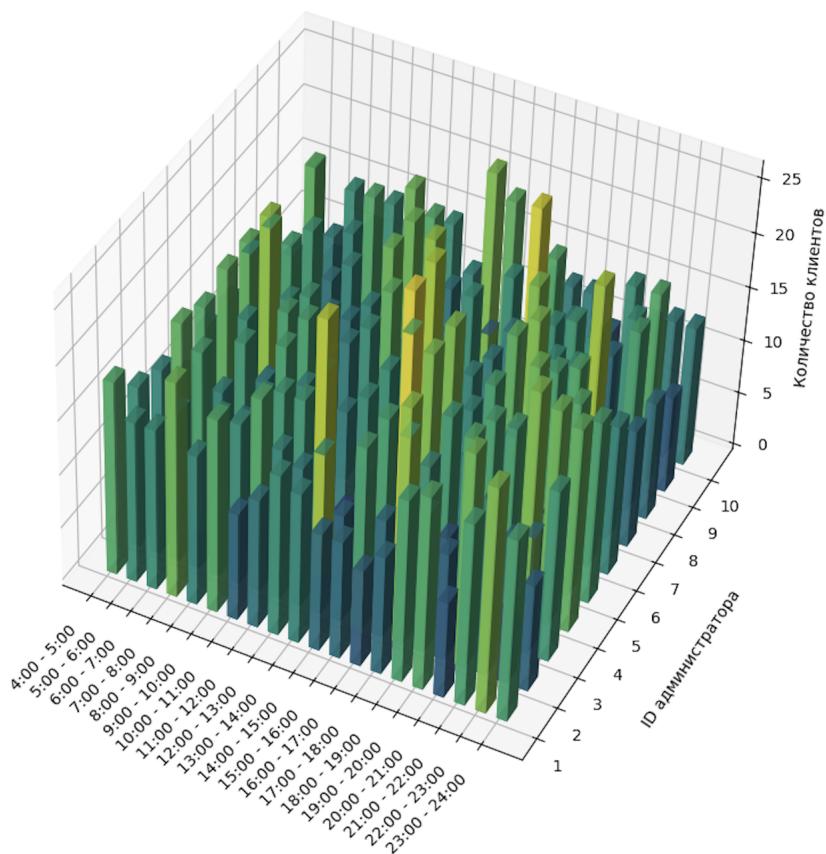


Рис. 38: Диаграмма результатов запроса 8

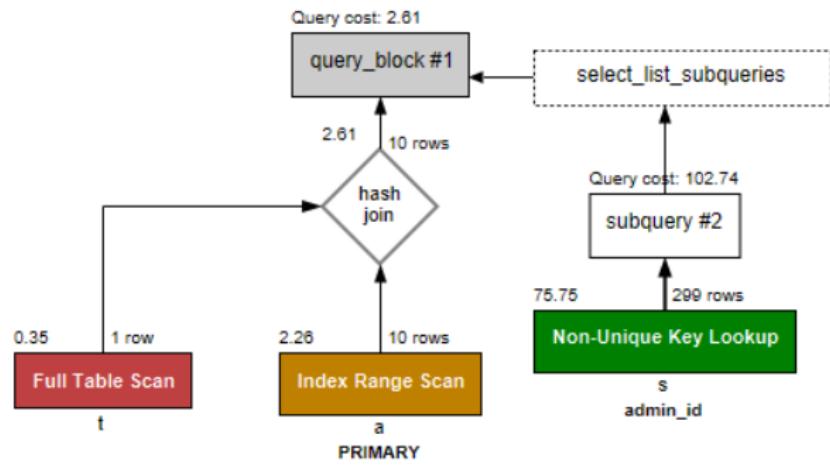


Рис. 39: Visual explain запроса 8

# Заключение

В рамках исследования был проведён анализ выбранной предметной области, после чего были составлены ER-диаграмма, схема объектов базы данных и таблицы метаданных. На их основе были созданы таблицы в MySQL, которые затем были заполнены с использованием программы на языке Python. После этого было выполнено восемь запросов к базе данных, некоторые из них представлены в 2 вариантах. Всего было создано 11 таблиц, в заполненной базе данных 309539 записей, на ER-диаграмме и схеме объектов базы данных 6 сущностей.

Работа выполнялась на устройстве MacBook Pro с чипом Apple M2 Pro и памятью 16ГБ, с использованиемialectа MySQL, среды программирования PyCharm, Python версии 3.1, командной строки и MySQL Workbench 8.0. Диаграммы результатов запросов были построены в PyCharm.

## Список литературы

- [1] Попов С. Г. Лист самостоятельной проверки хода выполнения курсовой работы по дисциплине «Теоретические основы баз данных» 4 семестр
- [2] - Маникюр [Электронный ресурс]: Википедия. URL: <https://ru.wikipedia.org/wiki/Маникюр> (Дата обращения: 12.05.2025)
- [3] Пилки - сеть салонов маникюра в СПб. URL: <https://pilkinail.ru/> (Дата обращения: 12.05.2025)
- [4] Takahashi M., Azuma S., Trend C. L. The manga guide to databases. – No Starch Press, 2009.
- [5] - Салон красоты [Электронный ресурс]: Википедия. URL: [https://ru.wikipedia.org/wiki/Салон\\_красоты](https://ru.wikipedia.org/wiki/Салон_красоты) (Дата обращения: 12.05.2025)
- [6] - Система управления базами данных [Электронный ресурс]: Википедия. URL: [https://ru.wikipedia.org/wiki/Система\\_управления\\_базами\\_данных](https://ru.wikipedia.org/wiki/Система_управления_базами_данных) (Дата обращения: 12.05.2025)

# Приложение А. Исходный код программы генерации схемы базы данных

```
1      CREATE DATABASE IF NOT EXISTS 'mybase' DEFAULT CHARACTER
2          SET utf8mb3;
3      USE 'mybase';
4
5      CREATE TABLE 'admin' (
6          'admin_id' int NOT NULL AUTO_INCREMENT,
7          'surname_id' int NOT NULL,
8          'name_id' int NOT NULL,
9          'patronymic_id' int NOT NULL,
10         PRIMARY KEY ('admin_id'),
11         KEY 'surname_id' ('surname_id'),
12         KEY 'name_id' ('name_id'),
13         KEY 'patronymic_id' ('patronymic_id'),
14         CONSTRAINT 'admin_ibfk_1' FOREIGN KEY ('surname_id')
15             REFERENCES 'surname' ('surname_id') ON DELETE CASCADE
16             ON UPDATE CASCADE,
17         CONSTRAINT 'admin_ibfk_2' FOREIGN KEY ('name_id')
18             REFERENCES 'name' ('name_id') ON DELETE CASCADE ON
19             UPDATE CASCADE,
20         CONSTRAINT 'admin_ibfk_3' FOREIGN KEY ('patronymic_id')
21             REFERENCES 'patronymic' ('patronymic_id') ON DELETE
22             CASCADE ON UPDATE CASCADE
23     ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=
24         utf8mb4_0900_ai_ci;
25
26      CREATE TABLE 'client' (
27          'client_id' int NOT NULL AUTO_INCREMENT,
28          'surname_id' int NOT NULL,
29          'name_id' int NOT NULL,
30          'patronymic_id' int NOT NULL,
31          PRIMARY KEY ('client_id'),
32          KEY 'surname_id' ('surname_id'),
33          KEY 'name_id' ('name_id'),
34          KEY 'patronymic_id' ('patronymic_id'),
35          CONSTRAINT 'client_ibfk_1' FOREIGN KEY ('surname_id')
36              REFERENCES 'surname' ('surname_id') ON DELETE CASCADE
37              ON UPDATE CASCADE,
38          CONSTRAINT 'client_ibfk_2' FOREIGN KEY ('name_id')
39              REFERENCES 'name' ('name_id') ON DELETE CASCADE ON
40              UPDATE CASCADE,
41          CONSTRAINT 'client_ibfk_3' FOREIGN KEY ('patronymic_id')
42              REFERENCES 'patronymic' ('patronymic_id') ON DELETE
43              CASCADE ON UPDATE CASCADE
44     ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=
45         utf8mb4_0900_ai_ci;
46
47      CREATE TABLE 'master' (
48          'master_id' int NOT NULL AUTO_INCREMENT,
```

```

34      'surname_id' int NOT NULL,
35      'name_id' int NOT NULL,
36      'patronymic_id' int NOT NULL,
37      'master_experience' int NOT NULL,
38      'master_age' int NOT NULL,
39      PRIMARY KEY ('master_id'),
40      KEY 'surname_id' ('surname_id'),
41      KEY 'name_id' ('name_id'),
42      KEY 'patronymic_id' ('patronymic_id'),
43      CONSTRAINT 'master_ibfk_1' FOREIGN KEY ('surname_id')
44          REFERENCES 'surname' ('surname_id') ON DELETE CASCADE
45          ON UPDATE CASCADE,
46      CONSTRAINT 'master_ibfk_2' FOREIGN KEY ('name_id')
47          REFERENCES 'name' ('name_id') ON DELETE CASCADE ON
48          UPDATE CASCADE,
49      CONSTRAINT 'master_ibfk_3' FOREIGN KEY ('patronymic_id')
50          REFERENCES 'patronymic' ('patronymic_id') ON DELETE
51          CASCADE ON UPDATE CASCADE
52 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=
53     utf8mb4_0900_ai_ci;

54
55     CREATE TABLE 'name' (
56         'name_id' int NOT NULL AUTO_INCREMENT,
57         'name' varchar(45) NOT NULL,
58         PRIMARY KEY ('name_id')
59     ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=
60         utf8mb4_0900_ai_ci;

61
62     CREATE TABLE 'okasusluga' (
63         'okasusluga_id' int NOT NULL AUTO_INCREMENT,
64         'service_id' int NOT NULL,
65         'schedule_id' int NOT NULL,
66         PRIMARY KEY ('okasusluga_id'),
67         KEY 'service_id' ('service_id'),
68         KEY 'schedule_id' ('schedule_id'),
69         CONSTRAINT 'okasusluga_ibfk_1' FOREIGN KEY ('service_id')
70             REFERENCES 'service' ('service_id') ON DELETE CASCADE
71             ON UPDATE CASCADE,
72         CONSTRAINT 'okasusluga_ibfk_2' FOREIGN KEY ('schedule_id'
73             ) REFERENCES 'schedule' ('schedule_id') ON DELETE
74             CASCADE ON UPDATE CASCADE
75     ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=
76         utf8mb4_0900_ai_ci;

77
78     CREATE TABLE 'patronymic' (
79         'patronymic_id' int NOT NULL AUTO_INCREMENT,
80         'patronymic' varchar(45) DEFAULT NULL,
81         PRIMARY KEY ('patronymic_id')
82     ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=
83         utf8mb4_0900_ai_ci;

84
85     CREATE TABLE 'schedule' (

```

```

72     'schedule_id' int NOT NULL AUTO_INCREMENT,
73     'schedule_date' date NOT NULL,
74     'time_id' int NOT NULL,
75     'schedule_free' tinyint(1) NOT NULL,
76     'master_id' int NOT NULL,
77     'admin_id' int NOT NULL,
78     'client_id' int NOT NULL,
79     PRIMARY KEY ('schedule_id'),
80     KEY 'master_id' ('master_id'),
81     KEY 'admin_id' ('admin_id'),
82     KEY 'client_id' ('client_id'),
83     KEY 'time_id' ('time_id'),
84     CONSTRAINT 'schedule_ibfk_1' FOREIGN KEY ('master_id')
85         REFERENCES 'master' ('master_id') ON DELETE CASCADE ON
86             UPDATE CASCADE,
87     CONSTRAINT 'schedule_ibfk_2' FOREIGN KEY ('admin_id')
88         REFERENCES 'admin' ('admin_id') ON DELETE CASCADE ON
89             UPDATE CASCADE,
90     CONSTRAINT 'schedule_ibfk_3' FOREIGN KEY ('client_id')
91         REFERENCES 'client' ('client_id') ON DELETE CASCADE ON
92             UPDATE CASCADE,
93     CONSTRAINT 'schedule_ibfk_4' FOREIGN KEY ('time_id')
94         REFERENCES 'time' ('time_id') ON DELETE CASCADE ON
95             UPDATE CASCADE
96 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=
97     utf8mb4_0900_ai_ci;

98     CREATE TABLE 'service' (
99     'service_id' int NOT NULL AUTO_INCREMENT,
100    'service_price' int NOT NULL,
101    'service_name' varchar(100) NOT NULL,
102    'service_duration' time NOT NULL,
103    PRIMARY KEY ('service_id')
104 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3;

105    CREATE TABLE 'surname' (
106    'surname_id' int NOT NULL AUTO_INCREMENT,
107    'surname' varchar(45) NOT NULL,
108    PRIMARY KEY ('surname_id')
109 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=
110     utf8mb4_0900_ai_ci;

111    CREATE TABLE 'time' (
112    'time_id' int NOT NULL AUTO_INCREMENT,
113    'time' time NOT NULL,
114    PRIMARY KEY ('time_id')
115 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=
116     utf8mb4_0900_ai_ci;

117    CREATE TABLE 'vozmusluga' (
118    'vozmusluga_id' int NOT NULL AUTO_INCREMENT,
119    'master_id' int NOT NULL,

```

```
113     'service_id' int NOT NULL,  
114     PRIMARY KEY ('vozmusluga_id'),  
115     KEY 'master_id' ('master_id'),  
116     KEY 'service_id' ('service_id'),  
117     CONSTRAINT 'vozmusluga_ibfk_1' FOREIGN KEY ('master_id')  
118         REFERENCES 'master' ('master_id') ON DELETE CASCADE ON  
119         UPDATE CASCADE,  
        CONSTRAINT 'vozmusluga_ibfk_2' FOREIGN KEY ('service_id')  
            REFERENCES 'service' ('service_id') ON DELETE CASCADE  
            ON UPDATE CASCADE  
    ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=  
        utf8mb4_0900_ai_ci;
```

## Приложение В. Исходный код программы за- полнения базы данных тестовыми данными

```
1 import mysql.connector
2 from mysql.connector import Error
3 import random
4 from datetime import datetime, timedelta
5
6 def create_connection():
7     connection = None
8     try:
9         connection = mysql.connector.connect(
10             host='localhost',
11             user='root',
12             password='sswthd2712',
13             database='mybase'
14         )
15
16     return connection
17
18 def insert_last_names(connection):
19     cursor = connection.cursor()
20     query = "INSERT INTO surname(surname) VALUES (%s)"
21
22     for i in range(1, 501):
23         last_name = f"Surname_{i}"
24
25     def insert_names(connection):
26         cursor = connection.cursor()
27         query = "INSERT INTO name(name) VALUES (%s)"
28
29         for i in range(1, 301):
30             name = f"Name_{i}"
31
32
33     def insert_patronymic(connection):
34         cursor = connection.cursor()
35         query = "INSERT INTO patronymic(patronymic) VALUES (%s)"
36
37         for i in range(1, 201):
38             patron = f"Patronymic_{i}"
39
40     def insert_service(connection, price, name, duration):
41         cursor = connection.cursor()
42         insert_query = "INSERT INTO service(service_price, "
43             "service_name, service_duration) VALUES (%s, %s, %s)"
44         values = (price, name, duration)
45
46     def random_time():
47         start = datetime.strptime("01:30", "%H:%M")
48         end = datetime.strptime("02:00", "%H:%M")
```

```

48     total_seconds = int((end - start).total_seconds())
49     random_seconds = random.randint(0, total_seconds)
50
51
52     random_time = start + timedelta(seconds=random_seconds)
53     return random_time.time()
54
55     a = create_connection()
56     insert_last_names(a)
57     insert_names(a)
58     insert_patronymic(a)
59
60     for i in range(101, 102):
61         price = random.randint(2000, 5000)
62         name = f"Service{i}"
63         duration = random_time()
64         insert_service(a, price, name, duration)
65
66         cursor = a.cursor()
67         cursor.execute("SELECT surname_id FROM surname")
68         surname = [row[0] for row in cursor.fetchall()]
69         cursor.execute("SELECT name_id FROM name")
70         name = [row[0] for row in cursor.fetchall()]
71         cursor.execute("SELECT patronymic_id FROM patronymic")
72         patronymic = [row[0] for row in cursor.fetchall()]
73
74         for _ in range(1, 201):
75             surname_id = random.choice(surname)
76             name_id = random.choice(name)
77             patronymic_id = random.choice(patronymic)
78
79             cursor.execute(
80                 "INSERT INTO client(surname_id, name_id, patronymic_id) "
81                 "VALUES (%s, %s, %s)",
82                 (surname_id, name_id, patronymic_id))
83         )
84         a.commit()
85
86         for _ in range(1, 51):
87             surname_id = random.choice(surname)
88             name_id = random.choice(name)
89             patronymic_id = random.choice(patronymic)
90
91             cursor.execute(
92                 "INSERT INTO admin(surname_id, name_id, patronymic_id) "
93                 "VALUES (%s, %s, %s)",
94                 (surname_id, name_id, patronymic_id))
95         )
96         a.commit()
97
98         for _ in range(1, 101):

```

```

98     surname_id = random.choice(surname)
99     name_id = random.choice(name)
100    patronymic_id = random.choice(patronymic)
101    opyt = random.randint(1, 10)
102    vosrast = random.randint(18, 50)
103
104    cursor.execute(
105        "INSERT INTO master(surname_id, name_id, patronymic_id, "
106        "master_experience, master_age) VALUES (%s, %s, %s, %s, "
107        "%s)",
108        (surname_id, name_id, patronymic_id, opyt, vosrast))
109    a.commit()
110
111    cursor = a.cursor()
112    cursor.execute("SELECT master_id FROM master")
113    masters = cursor.fetchall()
114    cursor.execute("SELECT COUNT(*) FROM service")
115    service_count = cursor.fetchone()[0]
116
117    for master in masters:
118        master_id = master[0]
119
120        num_services = random.randint(10, 20)
121
122        chosen_services = random.sample(range(1, service_count + 1), num_services)
123
124        for service_id in chosen_services:
125            insert_query = '''
126                INSERT INTO vozmusluga(master_id, service_id)
127                VALUES (%s, %s)
128            '''
129            cursor.execute(insert_query, (master_id, service_id))
130            a.commit()
131
132            cursor = a.cursor()
133            start_hour = 4
134
135            for i in range(20):
136                start_time = f"{start_hour+1}:00"
137                end_time = f"{start_hour+1+1}:00"
138                time_range = f"{start_time}-{end_time}"
139
140                cursor.execute(
141                    "INSERT INTO time(time) VALUES (%s)", (time_range,))
142                a.commit()
143
144                cursor = a.cursor()
145                cursor.execute("SELECT COUNT(*) FROM master")

```

```

147     master_count = cursor.fetchone()[0]
148     cursor.execute("SELECT COUNT(*) FROM admin")
149     admin_count = cursor.fetchone()[0]
150     cursor.execute("SELECT COUNT(*) FROM client")
151     client_count = cursor.fetchone()[0]
152
153     start_date = datetime.now()
154
155     for day in range(730):
156         schedule_date = start_date + timedelta(days=day)
157         for time_id in range(1, 21):
158             schedule_free = 0
159             master_id = random.randint(1, master_count)
160             admin_id = random.randint(1, admin_count)
161             client_id = random.randint(1, client_count)
162
163             query = '''
164             INSERT INTO schedule(schedule_date, time_id,
165             schedule_free, master_id, admin_id, client_id)
166             VALUES (%s, %s, %s, %s, %s, %s)
167             '''
168             data = (schedule_date.date(), time_id, schedule_free,
169                     master_id, admin_id, client_id)
170
171             cursor = a.cursor()
172             cursor.execute("SELECT schedule_id FROM schedule")
173             schedules = cursor.fetchall()
174             cursor.execute("SELECT COUNT(*) FROM service")
175             service_count = cursor.fetchone()[0]
176
177             for schedule in schedules:
178                 schedule_id = schedule[0]
179                 num_services = random.randint(15, 25)
180
181                 chosen_services = random.sample(range(1, service_count +
182                                         1), num_services)
183
184                 for service_id in chosen_services:
185                     insert_query = '''
186                     INSERT INTO okasusluga(service_id, schedule_id)
187                     VALUES (%s, %s)
188                     '''
189                     cursor.execute(insert_query, (service_id, schedule_id))
190                     a.commit()

```