

Федеральное государственное бюджетное образовательное учреждение высшего образования

# «МИРЭА – Российский технологический университет» РТУ МИРЭА

Институт КБ направление 10.05.05

Кафедра КБ-4 «Интеллектуальные системы информационной безопасности»

**КУРСОВАЯ РАБОТА**

**по дисциплине**

**«Клиент-серверные системы управления банком данных»**

Студент Мешков Артем Владимирович Ф.И.О.

подпись, дата

Группа ББСО-03-20

Работа защищена на оценку

Студент Мельников Николай Алексеевич Ф.И.О.

подпись, дата

Группа ББСО-03-20

Работа защищена на оценку

Руководитель работы Ф.И.О.

подпись, дата

Члены комиссии Ф.И.О.

подпись, дата

Ф.И.О.

подпись, дата

Москва 2023 г.

## **Оглавление**

Оглавление

[«МИРЭА – Российский технологический университет» РТУ МИРЭА 1](#_Toc144872320)

[**Оглавление** 2](#_Toc144872321)

[**Задание** 3](#_Toc144872322)

[**Введение** 4](#_Toc144872323)

[**Глава 1. База данных** 5](#_Toc144872324)

[Описание предметной области 5](#_Toc144872325)

[ER диаграмма 7](#_Toc144872326)

[Структура таблиц 8](#_Toc144872327)

[Автоматизация 16](#_Toc144872328)

[**Глава 2. Демонстрация работы** 20](#_Toc144872329)

[**Администратор** 20](#_Toc144872330)

[**Персонал** 22](#_Toc144872331)

[**Менеджер** 25](#_Toc144872332)

[**Аналитик** 27](#_Toc144872333)

[**Управляющий** 28](#_Toc144872334)

## **Задание**

В данной работе необходимо разработать базу данных для сети салонов красоты и облегчить процесс работы. После этого необходимо разработать интерфейс для удобного процесса взаимодействия пользователей с базой данных.

Цель базы данных должна состоять в управлении информацией о различных аспектах функционирования салона красоты. База данных должна быть предназначена для хранения и организации данных о заведении, сотрудниках, клиентах, услугах, заказах, чеках и пользователях.

## **Введение**

Главная цель базы данных заключается в обеспечении эффективного управления операциями, связанными с салоном красоты, а также обеспечении сохранности, целостности и доступности данных. База данных предоставляет возможность эффективно отслеживать и анализировать информацию о заведении, сотрудниках и клиентах, что помогает улучшать качество обслуживания и принимать обоснованные управленческие решения.

## **Глава 1. База данных**

### Описание предметной области

Данная предметная область рассматривает общие положения об управлении информацией о различных аспектах функционирования барбершопа, хранения и организации данных о заведении, сотрудниках, клиентах, услугах, заказах, чеках и пользователях. Заведение может иметь несколько сотрудников, каждый из которых может быть связан с одним или несколькими заведениями.

Предметная область подразумевает заказы услуг, а каждый заказ связан с определенным клиентом, услугой и мастером. Чеки содержат информацию о заказах и связаны с клиентом и заведением. Пользователи имеют доступ к системе и могут быть связаны с сотрудниками.

1. Управление информацией о заведении: База данных позволяет хранить данные о типе заведения, его адресе, почтовом индексе, номере телефона и количестве сотрудников. Это помогает в управлении и администрировании заведения, а также в организации связанных операций.
2. Управление информацией о сотрудниках: База данных содержит информацию о сотрудниках, их полных именах, должностях, контактной информации, опыте работы, заработной плате, краткой информации о них, возрасте и рейтинге. Это позволяет эффективно управлять кадровыми процессами, определять компетентность сотрудников и осуществлять мониторинг их производительности.
3. Управление информацией о клиентах: База данных хранит данные о клиентах, их полных именах, контактной информации и статусе клиента. Это позволяет управлять взаимодействием с клиентами, предоставлять персонализированные услуги и поддерживать связь с ними.
4. Управление информацией об услугах: База данных содержит данные об услугах, их названиях и ценах. Это помогает управлять предоставлением услуг, определением стоимости и обеспечением информированности клиентов о доступных услугах.
5. Управление информацией о заказах: База данных хранит данные о заказах, включая информацию о клиентах, услугах, датах заказов, рейтинге мастера и идентификаторе мастера. Это помогает организовать процесс заказа и управлять расписанием мастеров.
6. Управление информацией о чеках: База данных содержит данные о чеках, включая информацию о клиентах, заведении, заказе и стоимости. Это позволяет отслеживать и анализировать финансовые операции и обеспечивать прозрачность расчетов с клиентами.

Управление пользователями: База данных хранит данные о пользователях системы, их идентификаторах, именах, паролях, группах пользователей и идентификаторах сотрудников. Это обеспечивает контроль доступа, аутентификацию и авторизацию пользователей системы.

### ER диаграмма

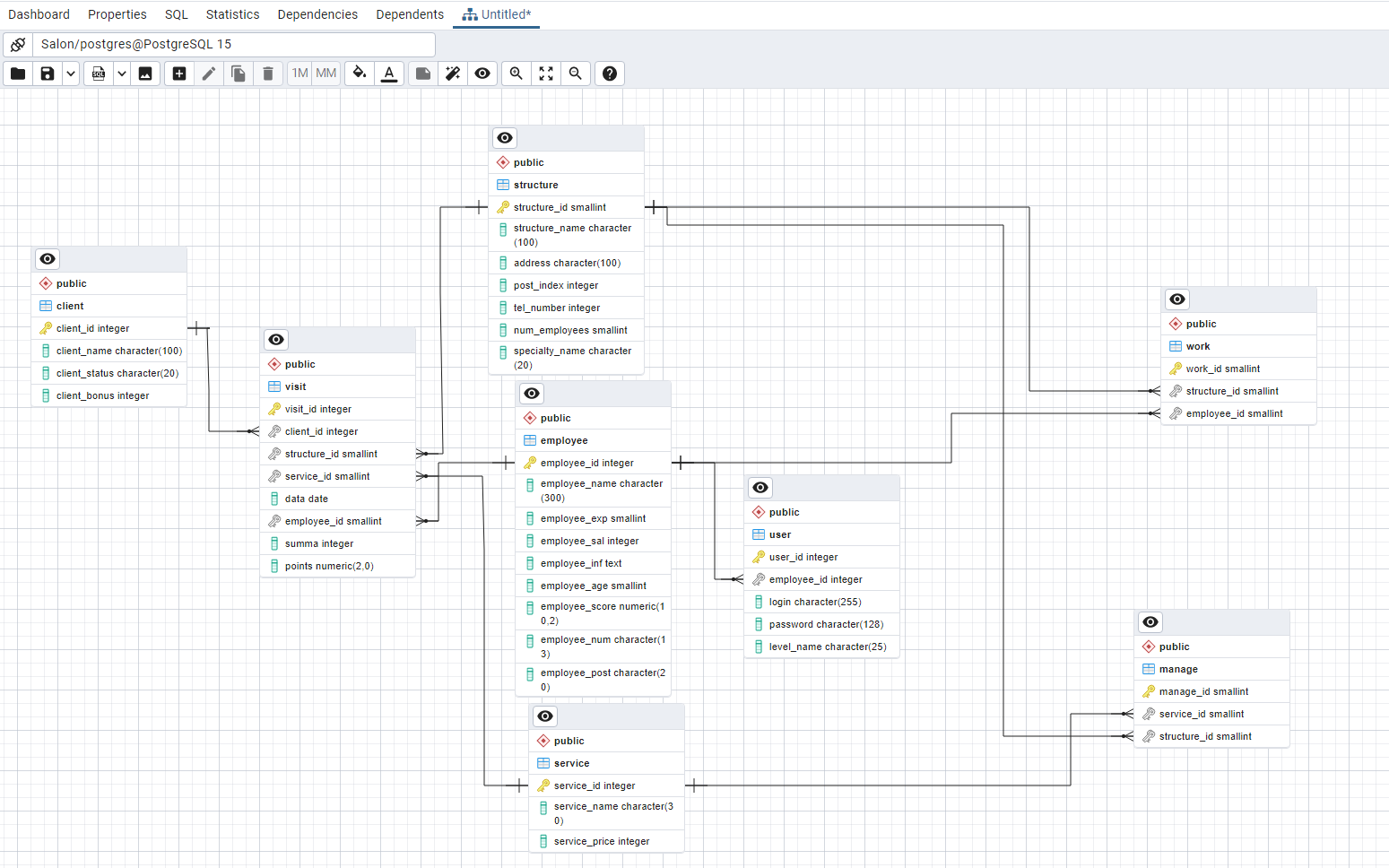
****

Рисунок 1 ER диаграмма

### Структура таблиц

Рассмотрим подробнее каждую таблицу

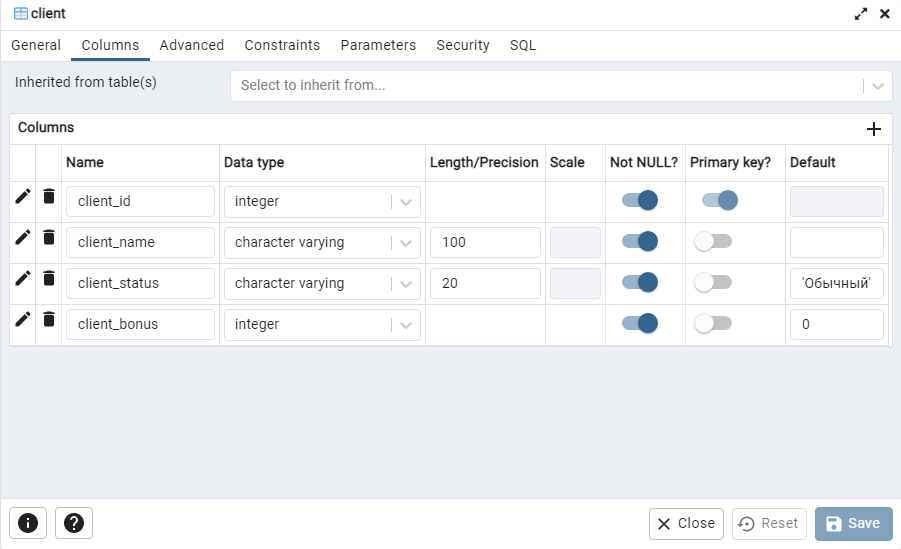
*  Таблица «Client»

Рисунок 2 Структура таблицы Client

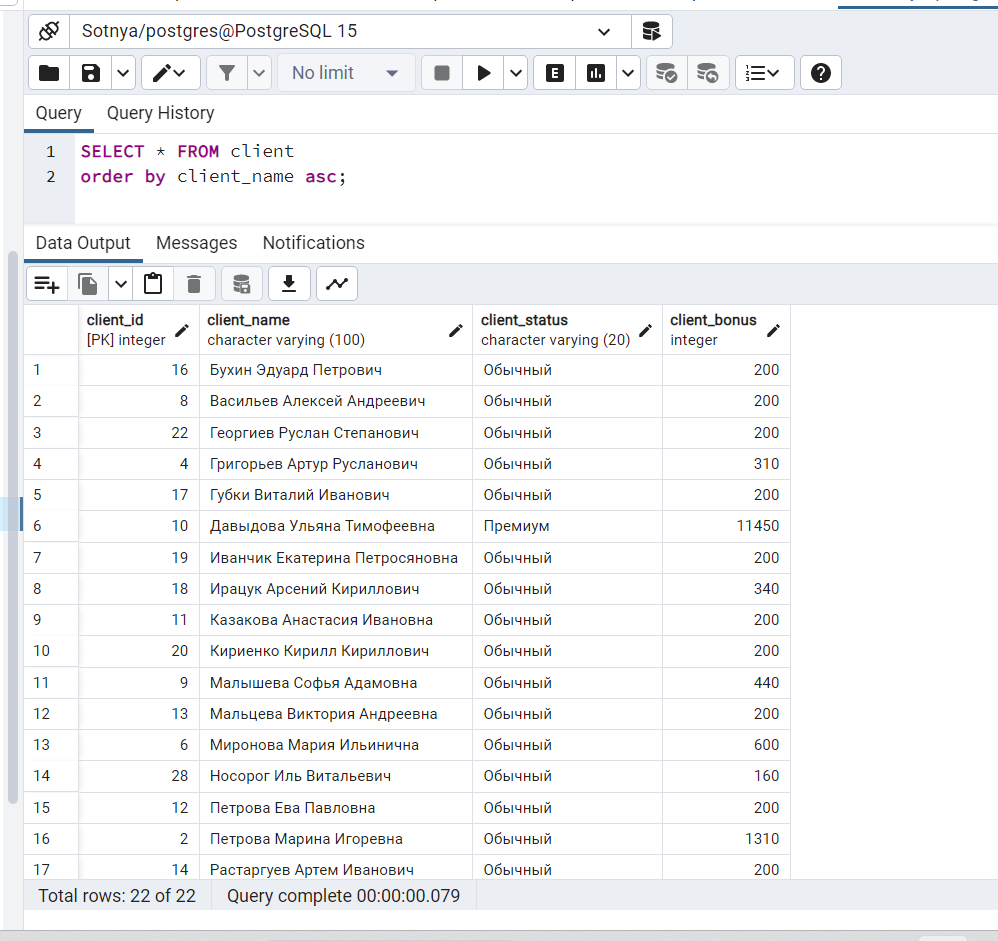
Данная таблица хранит информацию о клиенте. При добавлении нового клиента значения статуса и кол-во бонусов заполняются автоматически.

Рисунок 3 Наполнение таблицы Client

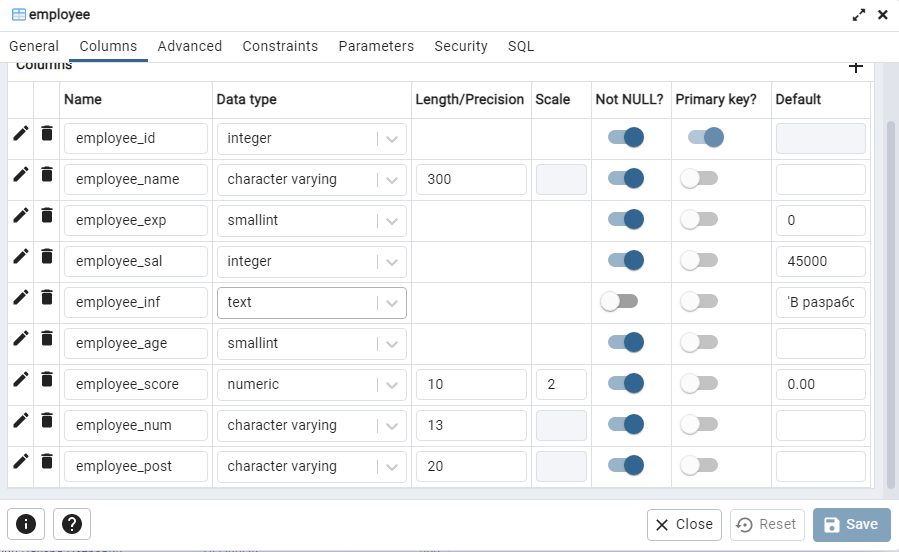
* Таблица «Employee»

Рисунок 4 Структура таблицы Employee

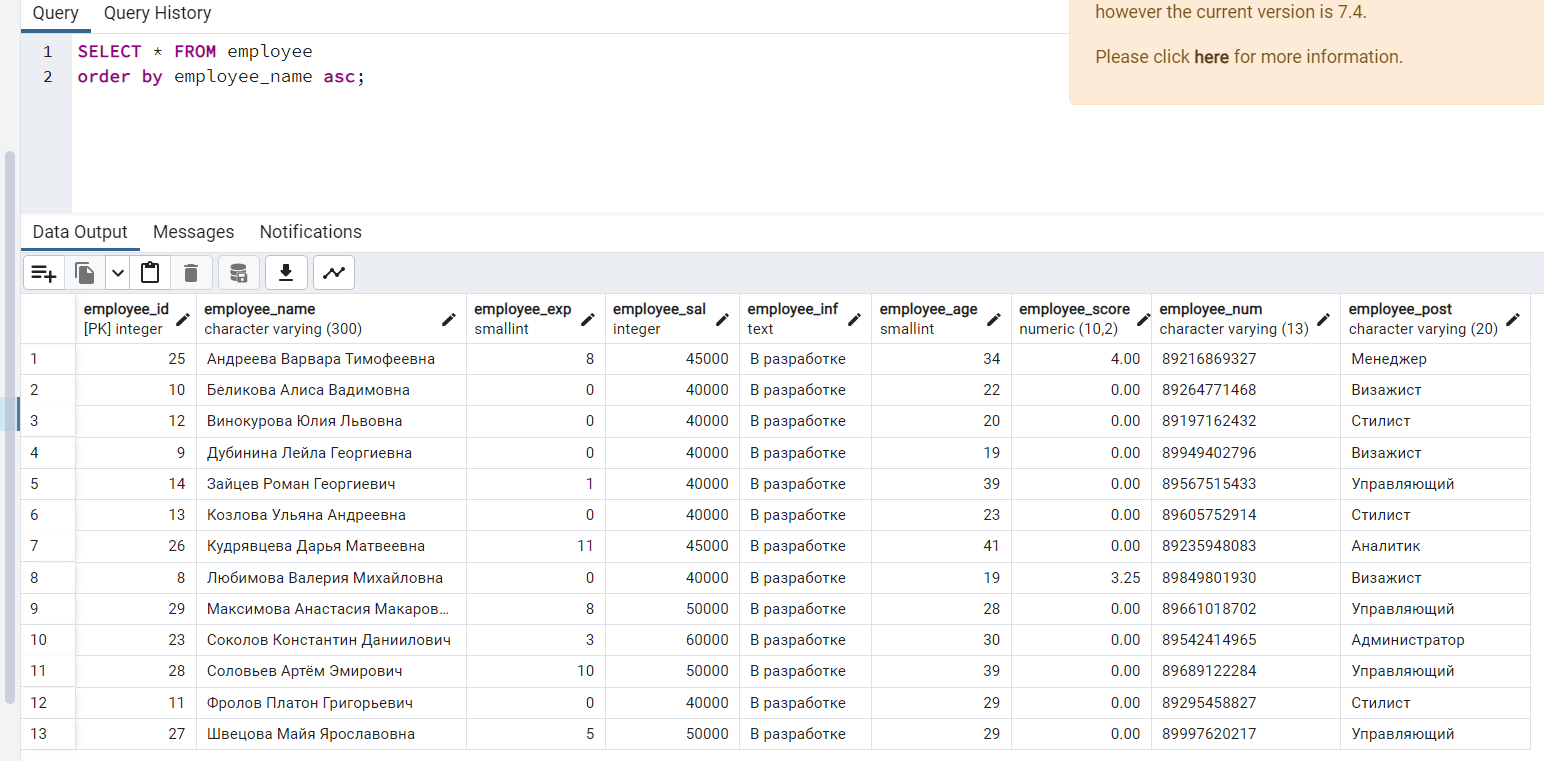
Данная таблица хранит информацию о сотрудниках. При добавлении нового сотрудника значения заработной платы, личной информации и рейтинге заполняются автоматически.

Рисунок 5 Наполнение таблицы Employee

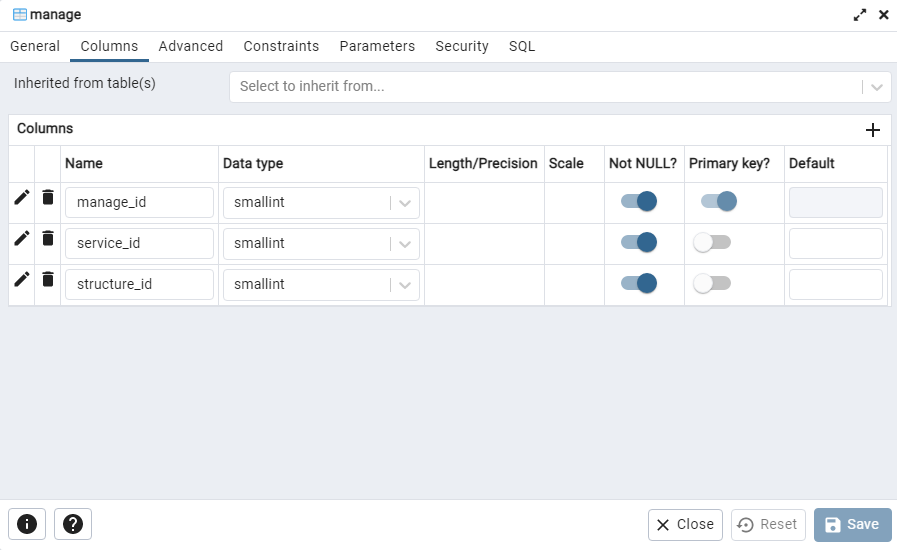
* Таблица «Manage»

Рисунок 6 Структура таблицы Manage

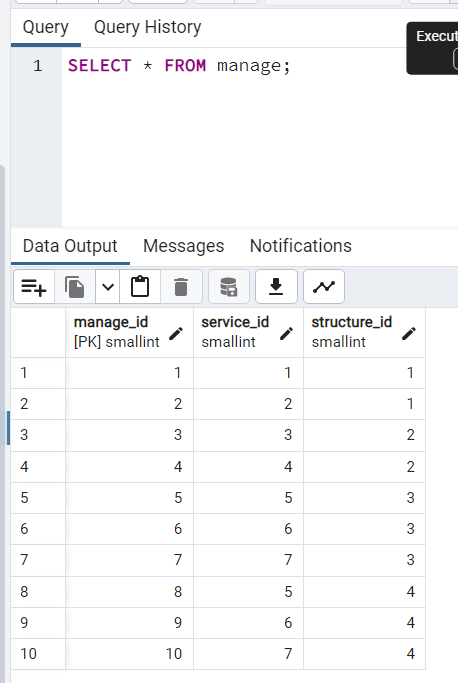
Данная таблица представляет связку двух таблиц. Она соединяет услуги с местом, в которых данная услуга предоставляется.

Рисунок 7 Наполнение таблицы Manage

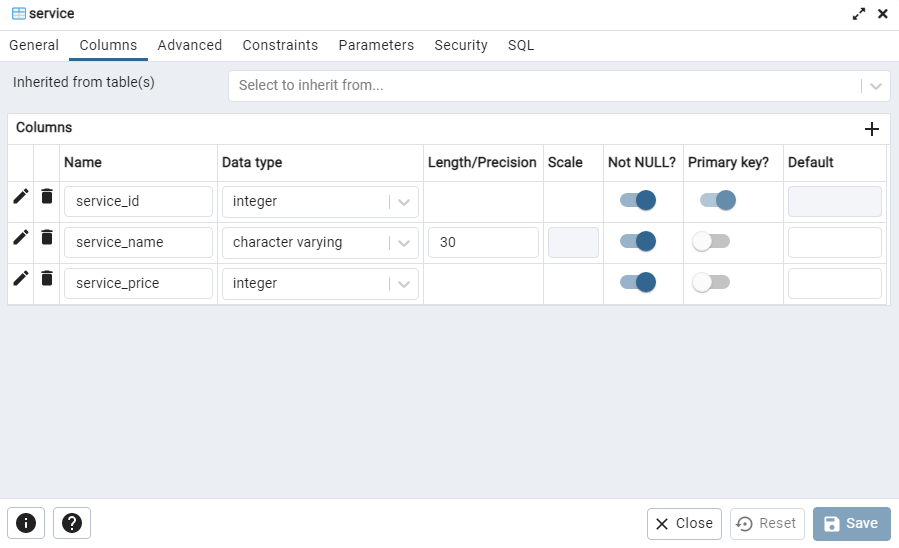
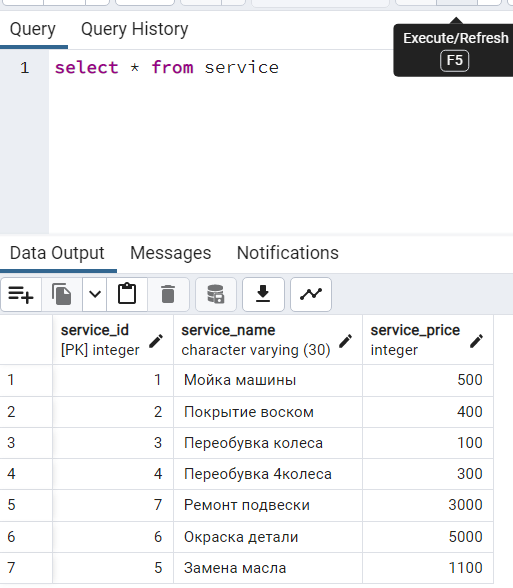
* Таблица «Service»

Рисунок 8 Структура таблицы Service

Данная таблица хранит информацию об услугах, а именно название и цену.

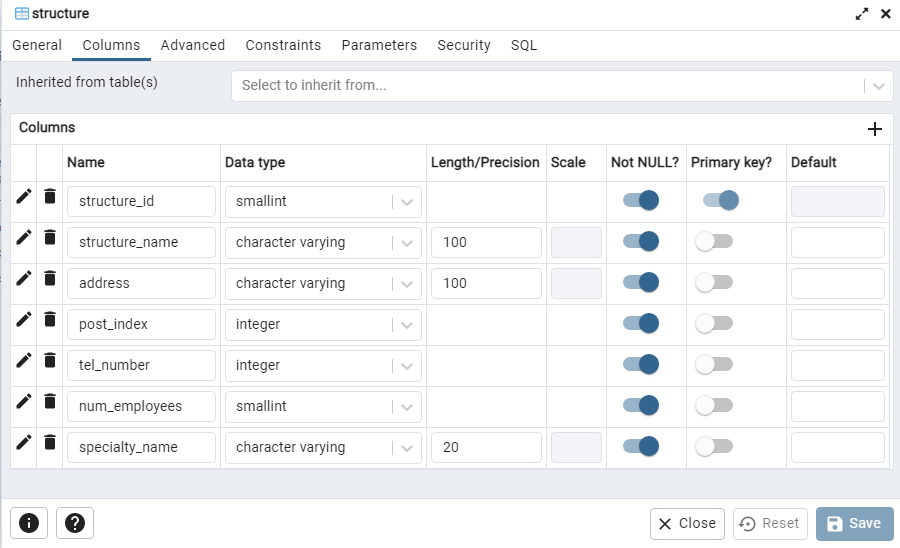
Таблица «Structure»

Рисунок 9 Наполнение таблицы Service

Рисунок 10 Структура таблицы Structure

Данная таблица хранит информацию о заведениях.

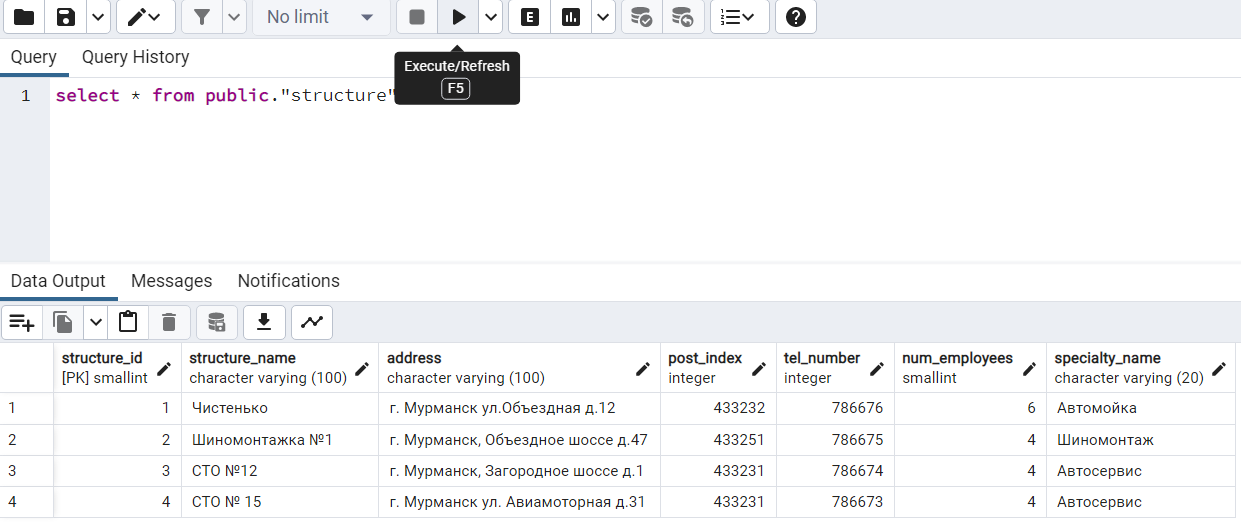


Рисунок 11 Наполнение таблицы Structure

* Таблица «User»

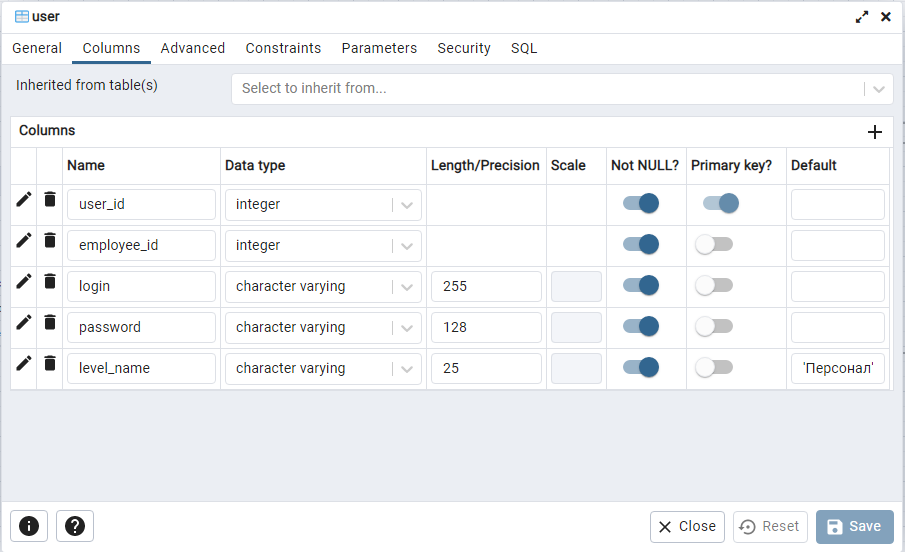


Рисунок 12 Структура таблицы User

Данная таблица хранит информацию о конкретных пользователях системы т.е. хранит логины и пароли, а также уровень доступа у данного пользователя. Заметим, что password занимает 128 символов, это сделано специально, ведь в этой колонке хранится пароль захешированный при помощи алгоритма sha-512 с солью.

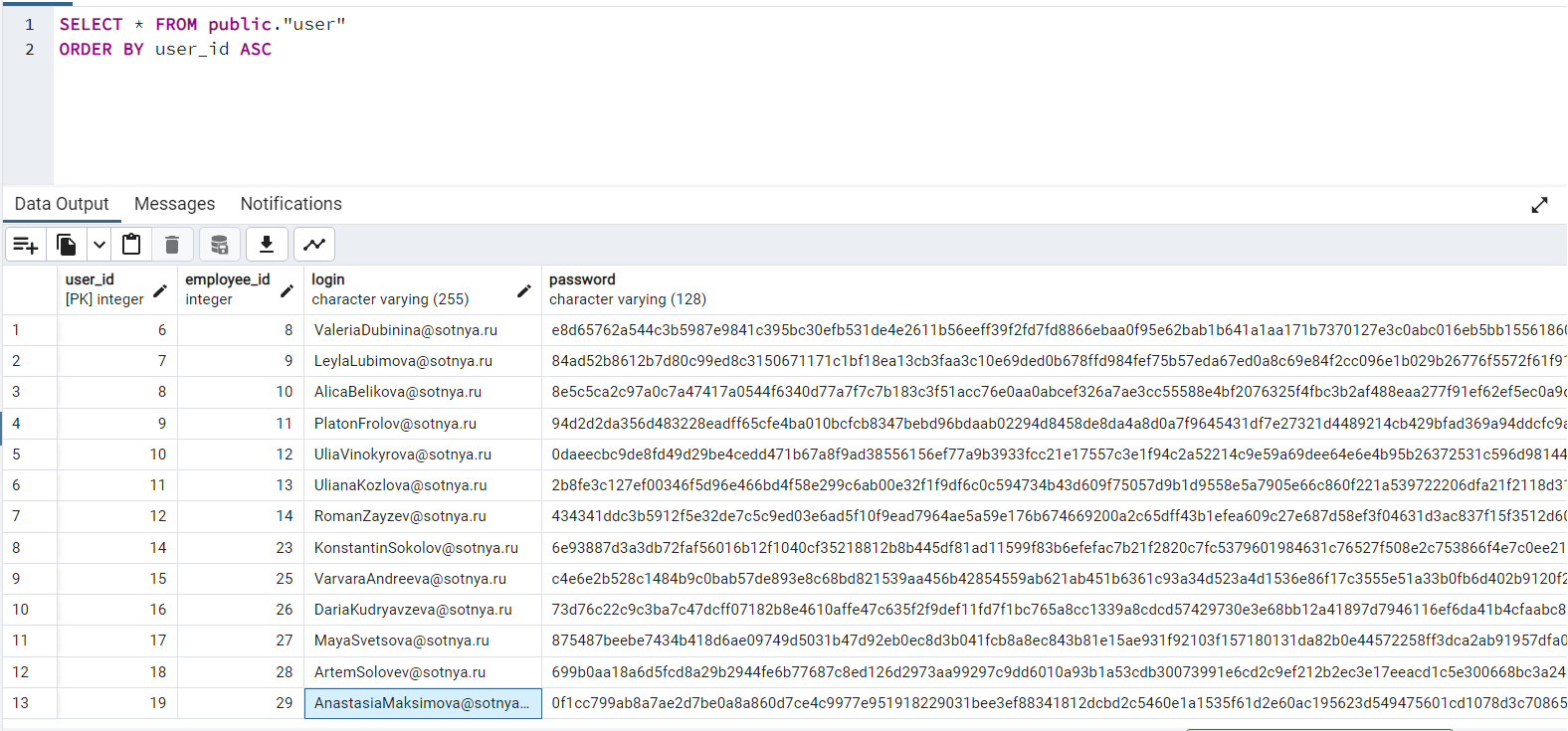


Рисунок 13 Наполнение таблицы User

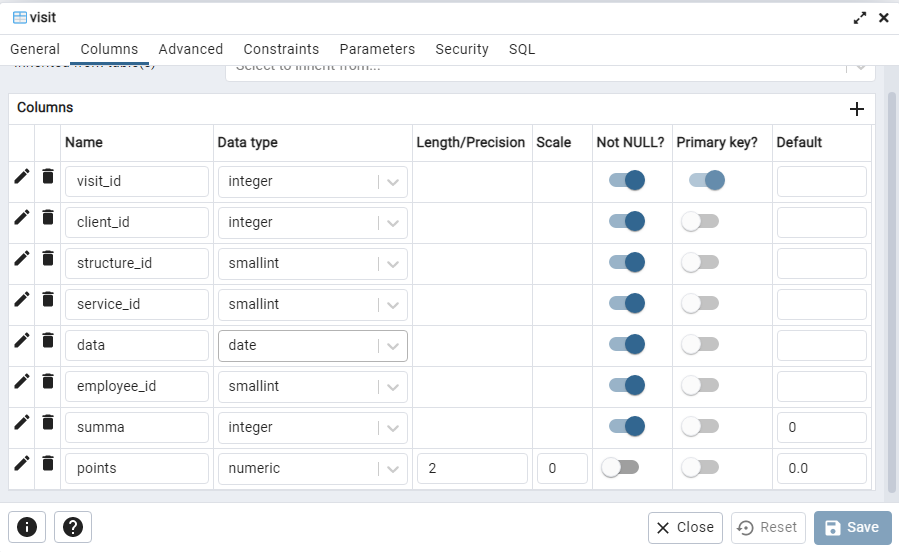
* Таблица «Visit»

Рисунок 14 Структура таблицы Visit

Данная таблица хранит информацию о посещениях.

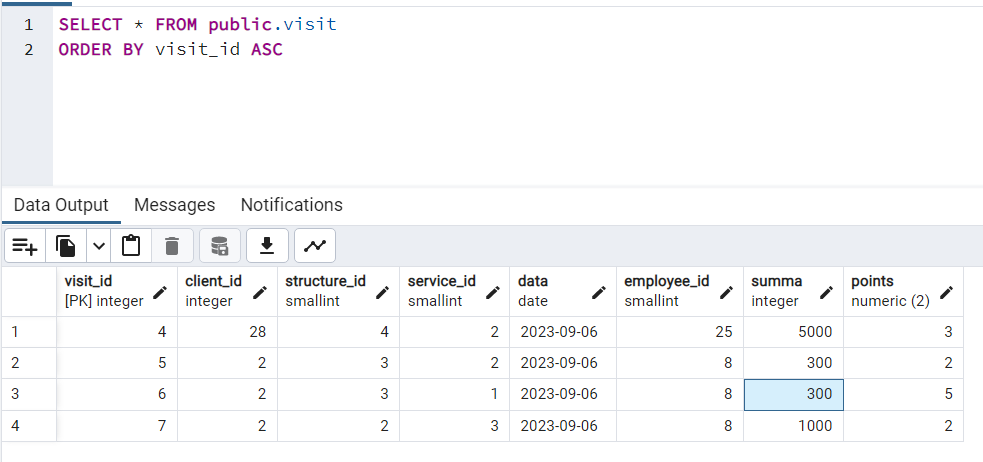


Рисунок 15 Наполнение таблицы Visit

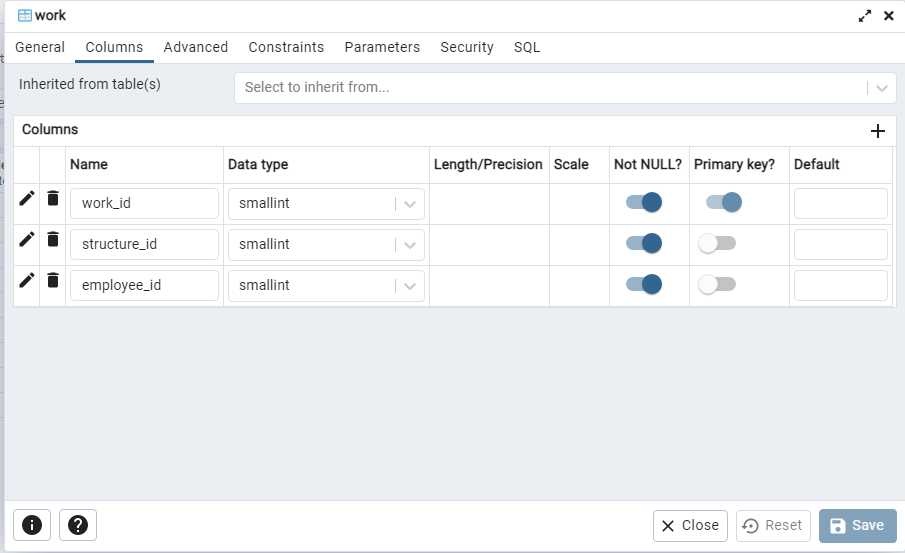
* Таблица «Work»

Рисунок 16 Структура таблицы Work

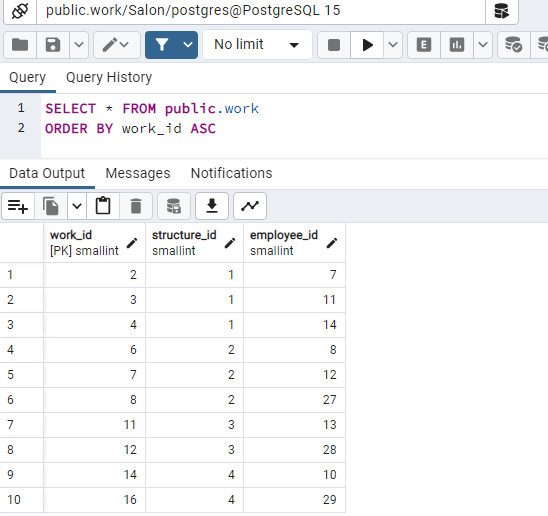
Данная таблица служит связкой для таблиц «Structure» и «Employee». Она необходимо чтобы понимать, в каком заведении работает конкретный сотрудник.

Рисунок 17 Наполнение таблицы Work

### Автоматизация

* Триггеры

Для автоматизации были использованы триггеры и триггерные функции

В нашем случае, необходимо создать 4 триггера, которые будут располагаться на таблице visit.

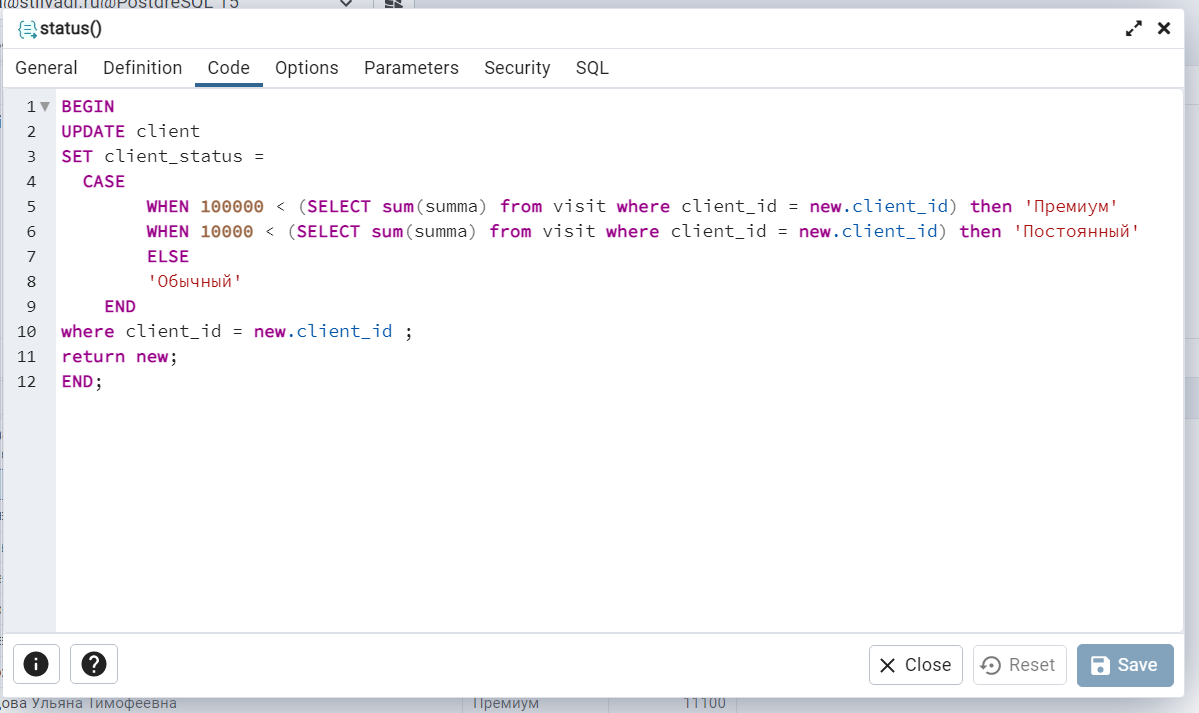
1-ый будет отвечать за статус клиента. В нашем случае если клиент потратил более 100.000 рублей, то он становится «Премиум», если потратил больше 10.000 рублей, то он становится «Постоянным»

Рисунок 18 Триггер №1

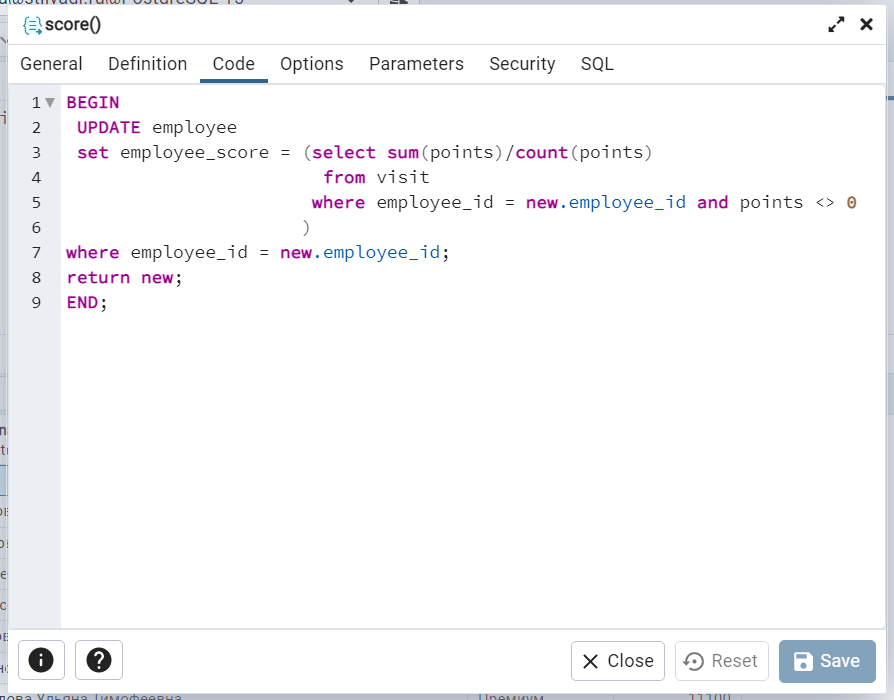
2-ой триггер будет отвечать за рейтинг парикмахеров. Он будет считать сумму всех не нулевых оценок и делить на их количество.

Рисунок 19 Триггер №2

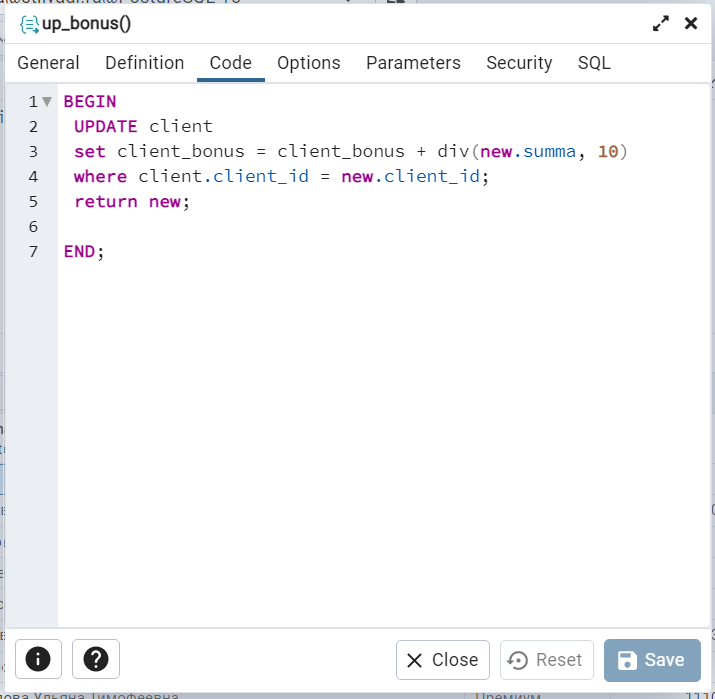
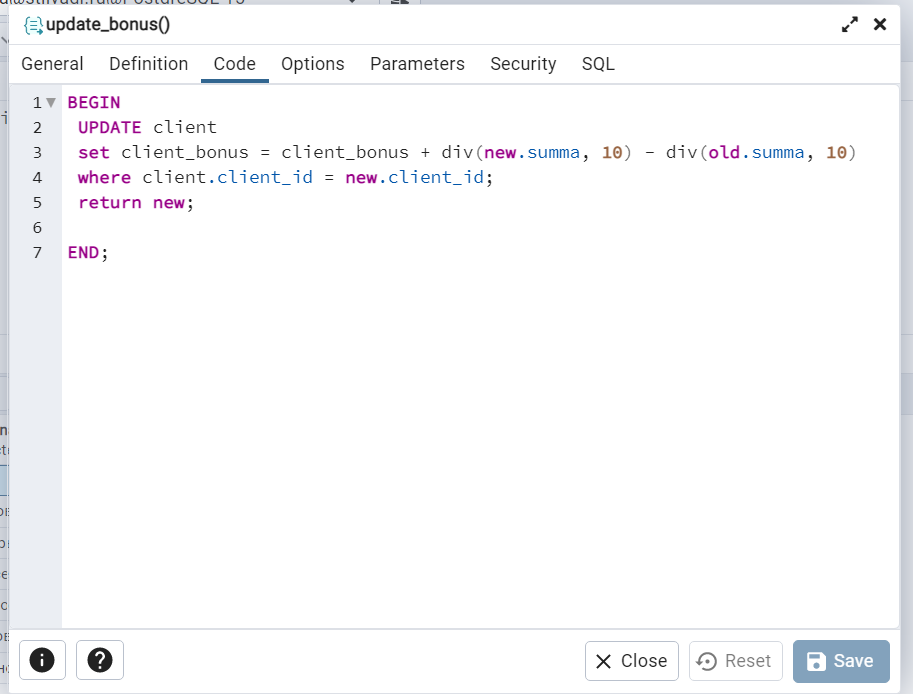
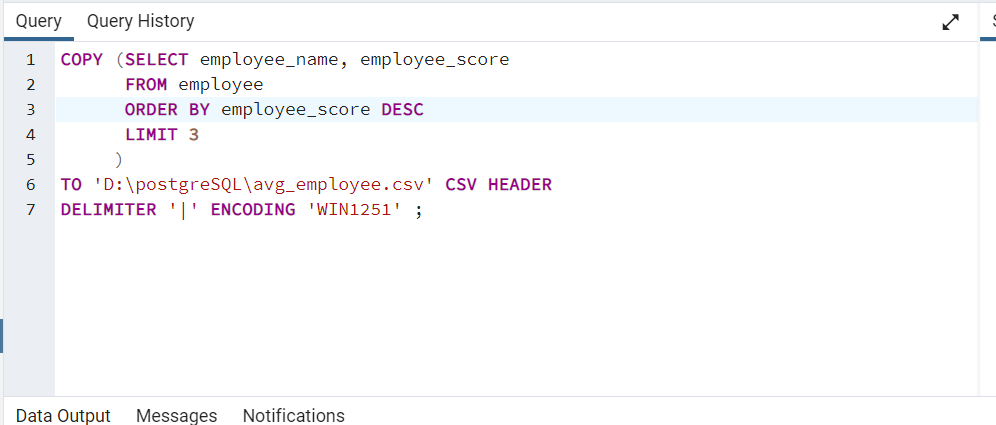
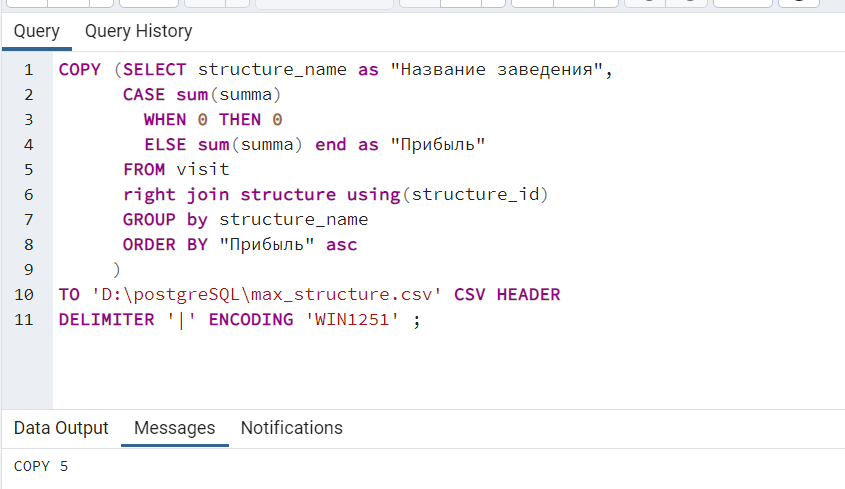
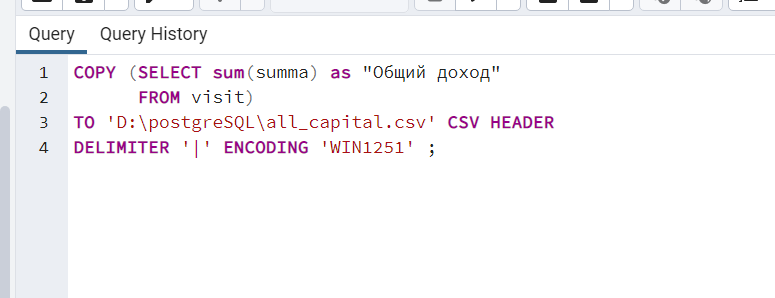
3-ий и 4-ый триггеры отвечают в целом за одно и тоже. Они считают бонусные рубли для каждого клиента. Бонусная программа - это 10% от суммы заказа. Разница лишь в том, что один работает на INSERT, а второй работает на UPDATE.

Рисунок 21 Триггер №4

Рисунок 20 Триггер №3



* Автоматические отчеты  
  + - Отчет о лучших сотрудниках месяца.
    - Отчет о прибыли каждого учреждения.
    - Общий доход заведений.

## **Глава 2. Демонстрация работы**

### **Администратор**

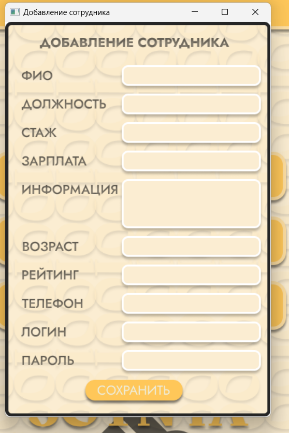


Рисунок 22 Добавление сотрудника

При нажатии на кнопку сохранить запускается sql скрипт, который добавляет сотрудника в таблицу employee, user и создает роль этому сотруднику в соответствии с его должностью

public static void addEmployee(String name, String position, String exp, String sal, String inf,String age,String score, String num, String login, String passSalt, String pass) throws SQLException, FileNotFoundException, ClassNotFoundException {  
 *statmt*.execute("INSERT INTO public.employee(\n" +  
 "\temployee\_name, employee\_exp, employee\_sal, employee\_inf, employee\_age, employee\_score, employee\_num, employee\_post)\n" +  
 "\tVALUES ('" + name +"', '" + exp + "', '" + sal + "', '" + inf + "', " + age + ", '" + score + "', " + num + ", '" + position +"');");  
 String employee\_id = *getEmployee\_byName*(name);  
 *statmt*.execute("INSERT INTO public.\"user\"(\n" +  
 "\temployee\_id, login, password)\n" +  
 "\tVALUES ('" + employee\_id + "', '" + login + "', '" + passSalt + "');");  
 *statmt*.execute("CREATE ROLE \""+ login +"\""+  
 "\tLOGIN\n" +  
 "\tNOSUPERUSER\n" +  
 "\tNOCREATEDB\n" +  
 "\tNOCREATEROLE\n" +  
 "\tINHERIT\n" +  
 "\tNOREPLICATION\n" +  
 "\tCONNECTION LIMIT -1\n" +  
 "\tPASSWORD '" + pass + "';");  
 String lvl="Staff";  
 if(position.equals("Администратор"))  
 lvl = "Administrator";  
 else if(position.equals("Аналитик"))  
 lvl = "Analyst";  
 else if(position.equals("Управляющий"))  
 lvl = "Local\_manager";  
 else if(position.equals("Менеджер"))  
 lvl = "Manager";  
 *statmt*.execute("GRANT \""+lvl+"\" TO \""+ login+"\" WITH ADMIN OPTION;");  
}

Стоит заметить, что при создании роли пароль передается в чисто виде (т.е. без хэширования и соли), а при занесении в таблицу user при помощи алгоритма sha512 с использованием соли. Защита от sql инъекции осуществляется при помощи jdbc. Для этого была добавлена соответсвующий драйвер, который работает с PostgreSQL.

### **Персонал**

Функционал у просто сотрудника это добавить клиента и добавить посещение. А также просмотр записей



Рисунок 23 Внешний вид Сотрудники

При вводе текста и нажатии на кнопку «Найти» вызывается функция, которая в двумерном массиве возвращает клиентов, удовлетворяющих поиску

public static String[][] getClient\_byNAME(String name) throws SQLException {  
  
 *data\_statmt* = *data\_connection*.createStatement();  
 String id = DataBase.*getLastClient*();  
 int kol = Integer.*parseInt*(id);  
 String[][] mas = new String[4][kol];  
 *data\_resSet* = *data\_statmt*.executeQuery("SELECT \* FROM client \n" +  
 "WHERE position('" + name + "' in client\_name)>0;");  
 int i = 0;  
 if(*data\_resSet* == null)  
 System.*out*.println(kol);  
 while(*data\_resSet*.next()) {  
 System.*out*.println("i/"+i);  
 mas[0][i] = *data\_resSet*.getString("client\_id");  
 mas[1][i] = *data\_resSet*.getString("client\_name");  
 mas[2][i] = *data\_resSet*.getString("client\_status");  
 mas[3][i] = *data\_resSet*.getString("client\_bonus");  
 i++;  
 }  
 return mas;  
}

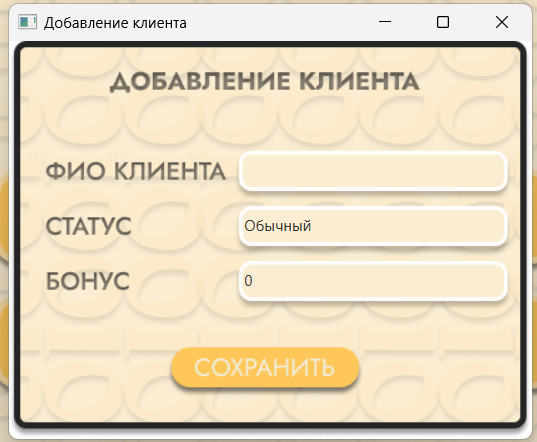
При нажатии на кнопку добавить клиента, появляется окно, в котором можно добавить клиента. При последующем нажатии на кнопку «Сохранить» данный клиент вноситься в базу путем вызова функции addClient

Рисунок 26 Добавление клиента

public static void addClient(String name) throws SQLException {  
 *data\_statmt*.execute("INSERT INTO public.client(\n" +  
 "\tclient\_name)\n" +  
 "\tVALUES ('" + name + "');");  
}

}

Также у персонала есть возможность добавить посещение путем нажатия на кнопку «добавить посещение» напротив конкретного клиента. После нажатия появляется новое окно, в котором при помощи выпадающего списка можно выбрать заведение и услугу.

### **Менеджер**

Рисунок 27 Редактирование посещения

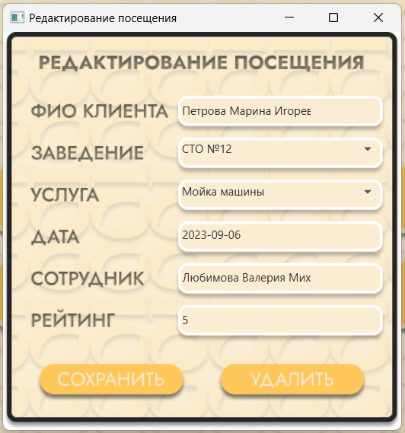
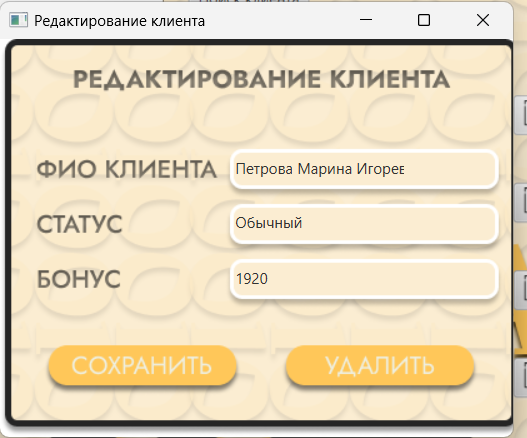
При нажатии на иконку редактирования открывается соответствующее окно, в котором уже появляется кнопка «удалить»

Рисунок 28 Редактирование клиента

Рисунок 27 Редактирование посещения



При нажатии на кнопки «удалить» вызывается функция delClient для удаления клиента. Удаление происходит каскадно, т.е. сначало удаляется из таблицы visit, а потом уде из таблицы client.

public static void delClient(String t1) throws SQLException {  
 *statmt*.execute("DELETE FROM visit "+  
 " where client\_id = " + t1+";");  
 *statmt*.execute("DELETE FROM client "+  
 " where client\_id = " + t1+";");  
}

При нажатии удалить в окне редактирования посещения вызывается функция delVisit.

public static void delVisit(String t1) throws SQLException {  
 *statmt*.execute("DELETE FROM visit "+  
 " where visit\_id = " + t1+";");  
}

### **Аналитик**

Аналитик по условию может просматривать все страницы

Рисунок 29 Главный экран у аналитика

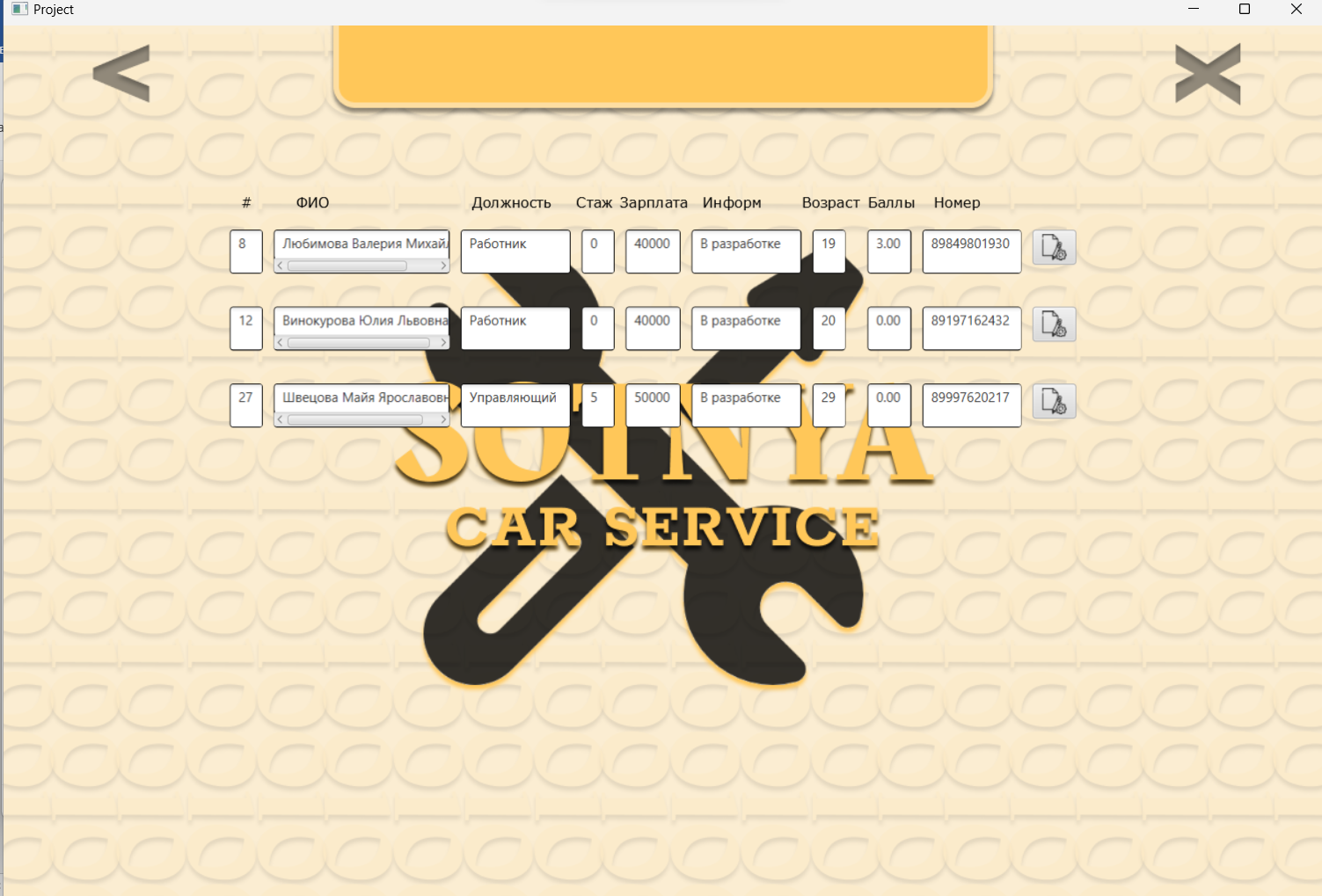


А также смотреть отчеты, которые расположены в соответствующем месте. Отчеты при помощи скрипта сохраняются на жестком диске.

### **Управляющий**

Последним остался управляющий, в его полномочиях редактировать информацию о заведениях и сотрудниках.

Рисунок 30 Информация о своих сотрудниках



При нажатии на кнопку редактирования открывается новое окно, в котором управляющий может поправить информацию о соответствующем сотруднике

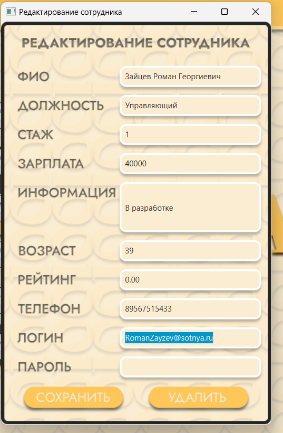


Рисунок 31 Редактирование сотрудника

При нажатии на кнопку данные обновляются вне зависимости от того, изменились они или нет. Данные обновляются путем вызова функции UpdateEmployee

public static void UpdateEmployee(String id, String t1, String t2, String t3, String t4,  
 String t5, String t6, String t7, String t8) throws SQLException {  
 *data\_statmt*.execute("UPDATE employee " +  
 "set employee\_name = '" + t1 + "'," +  
 " employee\_post = '"+ t2 + "'," +  
 " employee\_exp = " + t3 + "," +  
 " employee\_inf = '" + t5 + "'," +  
 " employee\_age = " + t6 + "," +  
 " employee\_num = " + t8 +  
 " where employee\_id = " + id+";");  
}

Рисунок 32 Просмотр своего заведения

При нажатии на кнопку «сохранить» вызывается метод UpdateStructure.

public static void UpdateStructure(String id, String t1, String t2, String t3, String t4, String t5, String t6) throws SQLException {  
 //System.out.println(t1+"|"+t2+"|"+ t3+"|" + t4);  
 *statmt*.execute("UPDATE structure " +  
 "set structure\_name = '" + t1 + "'," +  
 " address = '"+ t2 + "'," +  
 " specialty\_name = '" + t3 + "'," +  
 " post\_index = '" + t4 + "'," +  
 " tel\_number = '" + t5 + "'," +  
 " num\_employees = " + t6 +  
 " where structure\_id = " + id+";");  
}



Рисунок 32 Отображение заведения

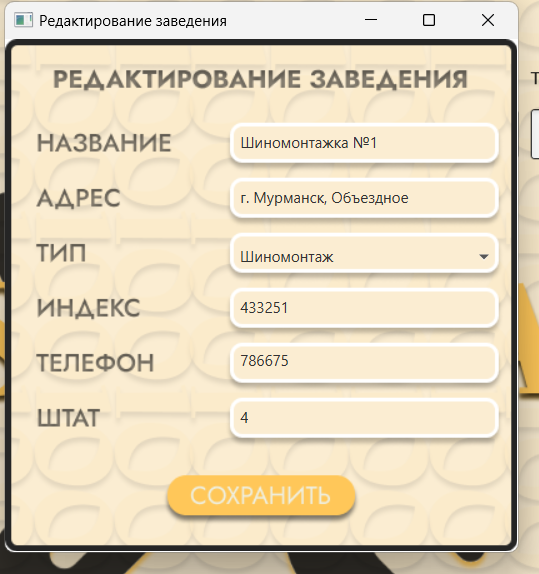


Рисунок 33 Редактирование своего заведения

**Список источников**

"PostgreSQL Documentation" // PostgreSQL [Электронный ресурс] URL: <https://www.postgresql.org/docs/14/index.html>

"PostgreSQL Tutorial" // PostgreSQL Tutorial [Электронный ресурс] URL:<https://www.postgresqltutorial.com/>

"JavaFX Documentation" // OpenJFX [Электронный ресурс] URL:<https://openjfx.io/>

"JavaFX Tutorial" // Tutorialspoint [Электронный ресурс] URL:<https://www.tutorialspoint.com/javafx/index.htm>

"JavaMail API Documentation" // JavaEE [Электронный ресурс] URL:<https://javaee.github.io/javamail/>

"PostgreSQL JDBC Driver Documentation" // PostgreSQL JDBC Driver [Электронный ресурс] URL:<https://jdbc.postgresql.org/documentation/>