Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(МОСКОВСКИЙ ПОЛИТЕХ)

**КУРСОВОЙ ПРОЕКТ**

По курсу **Проектирование и администрирование баз данных**

по направлению 09.03.01 − Информатика и вычислительная техника

Образовательная программа (профиль) «Системная и программная инженерия»

ТЕМА

**«Проектирование и разработка базы данных «ПРОКАТ АВТОМОБИЛЕЙ» в СУБД PostgreSQL»**

Студент: \_\_\_\_\_\_\_\_\_\_\_\_ / Абдулаева Камилла Дамировна 221–3210/

Проверил: \_\_\_\_\_\_\_\_\_\_ к.т.н., доцент Евдошенко О.И.

Москва, 2023

СОДЕРЖАНИЕ

[ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ 3](#_Toc139233775)

[ВВЕДЕНИЕ 4](#_Toc139233776)

[1 Описание предметной области 5](#_Toc139233777)

[2 Проектирование и разработка базы данных 7](#_Toc139233778)

[2.1 Концептуальное проектирование базы данных 7](#_Toc139233779)

[2.2 Логическое проектирование базы данных 11](#_Toc139233780)

[2.3 Выбор и описание СУБД 13](#_Toc139233781)

[2.4 Физическое проектирование базы данных 15](#_Toc139233782)

[3 Использование базы данных 26](#_Toc139233783)

[3.1 Создание DataSet 26](#_Toc139233784)

[3.2 Добавление данных в базу 27](#_Toc139233785)

[3.3 Запросы к базе данных 28](#_Toc139233786)

[3.4 Тестирование функций и триггеров 44](#_Toc139233787)

[ЗАКЛЮЧЕНИЕ 60](#_Toc139233788)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 61](#_Toc139233789)

Федеральное государственное автономное

образовательное учреждение высшего образования

**«Московский политехнический университет»**

**Кафедра:** Инфокогнитивные технологии

**Направление подготовки:** 09.03.01 “Информатика и вычислительная техника”, направленность (профиль): Системная и программная инженерия

**Дисциплина:** Проектирование и администрирование баз данных

**Тема:** Проектирование и разработка базы данных «Прокат автомобилей» в СУБД PostgreSQL

# ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

1. Необходимо выбрать и проанализировать предметную область.
2. Определить сущности (>5), атрибуты (>3), связи между сущностями. Привести отношения к 3 НФ.
3. Разработать базу данных средствами СУБД на основе спроектированной модели. Установить связи для поддержки ссылочной целостности.
4. Разработать SQL-запросы (DML) (не менее 20) различной степени сложности (агрегатные функции, группировка (GROUP BY, HAVING), подзапросы, CASE, оконные функции)
5. Разработать функции для расчетов (3 функции, используя язык SQL и plpgSQL)
6. Разработать триггеры (5 триггеров)

**Исполнитель**: студент гр. 221–3210 Абдулаева К.Д

подпись

**Руководитель**: к.т.н., доцент кафедры ИКТ Евдошенко О. И.

подпись

# **ВВЕДЕНИЕ**

Предметная область данной работы связана с коммерческой деятельностью, а именно - прокатом автомобилей. Автомобильный прокат является одной из самых популярных услуг в современном мире. Клиенты могут арендовать автомобиль на любой период времени и наслаждаться своей поездкой. Однако, для успешной работы пункта проката автомобилей необходимо вести учет и контролировать все процессы, что и будет реализовано в ходе выполнения.

В результате работы будет создана база данных на основе СУБД PostgreSQL, содержащая структуру таблиц, которая будет учитывать стоимость проката автомобилей, типы автомобилей, годы выпуска и многие другие параметры. Также будет разработана система штрафов и скидок для клиентов, механизм отслеживания всех транзакций и данных клиентов.

**Цель работы:**

* Систематизация и закрепление полученных теоретических знаний и практических умений по разработке баз данных с использованием СУБД PostgreSQL.

**Задачи работы:**

* Изучить литературу по проектированию и разработке базы данных в СУБД PostgreSQL;
* Получить практический опыт разработки базы данных в СУБД PostgreSQL.

Основанием для выполнения курсовой работы является учебный план дисциплины «Проектирование и администрирование баз данных», нацеленный на приобретение новых и закрепление уже имеющихся знаний и навыков в разработке структур баз данных.

# **1 Описание предметной области**

Предметной областью данной информационной системы является автоматизация процессов проката автомобилей. Основные бизнес-процессы представлены в [Таблице 1.](#таблица1)

[Таблица 1 - Основные бизнес-процессы](#таблица1текст)

|  |  |
| --- | --- |
| Номер бизнес-процесса | Наименование бизнес-процесса |
| 1. | Обновление данных по автомобилям |
| 2. | Работа с ценами на прокат автомобилей |
| 3. | Работа со скидками для клиентов |
| 4. | Внесение изменений в таблице «Аренды» |
| 5. | Обработка штрафов |

1. Обновление данных по автомобилям

В рамках данного бизнес-процесса добавляются новые модели автомобилей в автопарке, либо меняется количество уже существующих.

1. Работа с ценами на прокат автомобилей

В рамках данного бизнес-процесса происходит расчет стоимости проката автомобиля, где факторами формирования стоимости являются

1. Дата выпуска транспортного средства
2. Длительность проката
3. Работа со скидками для клиентов

В рамках данного бизнес-процесса происходит расчет скидки для постоянных клиентов и добавление клиента в таблицу постоянных клиентов.

1. Внесение изменений в таблице «Аренды»

В рамках данного бизнес-процесса происходит обработка заказов на прокат автомобилей, внесение данных о каждом прокате.

1. Обработка штрафов

В рамках данного бизнес-процесса происходит формирование штрафов за:

1. Просрочку даты возврата автомобиля
2. Возврат транспортного средства в ненадлежащем виде

Основными задачами информационной системы являются автоматизация процессов проката автомобилей, повышение эффективности работы и рационального использования данных автопарка.

# **2 Проектирование и разработка базы данных**

База данных — это хранилище информации, которое организовано таким образом, чтобы облегчить хранение, поиск и обработку данных. Проектирование и создание базы данных — это важный этап в разработке системы управления данными, который позволяет создать эффективную и надежную систему хранения и обработки информации.

## 2.1 Концептуальное проектирование базы данных

Концептуальное проектирование БД — это начальный этап разработки базы данных, в рамках которого строится ее логическая структура и создается концептуальная модель БД. В этом процессе определяются типы данных, атрибуты сущностей, связи между ними и их характеристики.

Концептуальное проектирование БД требует понимания всех бизнес-процессов, которые база данных должна обслуживать. Важно понять, какие данные необходимо хранить, как они будут использоваться и как связаны между собой.

При концептуальном проектировании БД используются стандартные способы моделирования, такие как ER-диаграммы, которые помогают визуализировать сущности и связи между ними в БД. Это позволяет убедиться, что все необходимые данные учтены и правильно представлены, и создать представление о том, как они будут использоваться при построении приложений или работе с данными.

Структура сущности «Аренды» описывается в [Таблице 2](#таблица2).

[Таблица 2 - Структура сущности «Аренды»](#таблица2текст)

|  |  |  |  |
| --- | --- | --- | --- |
| Название параметра | Тип данных | Размер | Диапазон значений |
| Автомобиль | целочисленный | 4 байта | Положительные целые числа |
| Клиент | целочисленный | 4 байта | Положительные целые числа |
| Дата выдачи | дата | 4 байта | ДД.ММ.ГГ |
| Дата возврата | дата | 4 байта | ДД.ММ.ГГ |

Продолжение [Таблицы 2.](#таблица2текст)

|  |  |  |  |
| --- | --- | --- | --- |
| Название параметра | Тип данных | Размер | Диапазон значений |
| Счет | число двойной точности | 8 байт | Положительные числа |
| Заказ | целочисленный | 4 байта | Уникальный идентификатор, положительные целые числа |
| Состояние автомобиля | целочисленный | 4 байта | Значение от 1 до 5 |
| Реальная дата возврата | дата | 4 байта | ДД.ММ.ГГ |

Структура сущности «Автомобили» описывается в [Таблице 3.](#таблица3)

[Таблица 3 - Структура сущности «Автомобили»](#таблица3текст)

|  |  |  |  |
| --- | --- | --- | --- |
| Название параметра | Тип данных | Размер | Диапазон значений |
| Автомобиль | целочисленный | 4 байта | Уникальный идентификатор, положительные целые числа |
| Марка автомобиля | целочисленный | 4 байта | Положительные целые числа |
| Стоимость автомобиля | целочисленный | 4 байта | Положительные целые числа |
| Стоимость аренды | целочисленный | 4 байта | Положительные целые числа |
| Тип автомобиля | символьный | 255 байт | Текстовые символы |
| Дата выпуска | дата | 4 байта | ДД.ММ.ГГ |
| Полное название | символьный | 255 байт | Текстовые символы |
| Количество | целочисленный | 4 байта | Положительные целые числа |

Структура сущности «Клиенты» описывается в [Таблице 4.](#таблица4)

[Таблица 4 - Структура сущности «Клиенты»](#таблица4текст)

|  |  |  |  |
| --- | --- | --- | --- |
| Название параметра | Тип данных | Размер | Диапазон значений |
| Клиент | целочисленный | 4 байта | Уникальный идентификатор, положительные целые числа |
| Фамилия | символьный | 255 байт | Текстовые символы |
| Имя | символьный | 255 байт | Текстовые символы |

Продолжение [Таблицы 4](#таблица4текст).

|  |  |  |  |
| --- | --- | --- | --- |
| Название параметра | Тип данных | Размер | Диапазон значений |
| Отчество | символьный | 255 байт | Текстовые символы |
| Адрес | символьный | 255 байт | Текстовые символы |
| Номер телефона | символьный | 255 байт | Текстовые символы |

Структура сущности «Постоянные клиенты» описывается в [Таблице 5.](#таблица5)

[Таблица 5 - Структура сущности «Постоянные клиенты»](#таблица5текст)

|  |  |  |  |
| --- | --- | --- | --- |
| Название параметра | Тип данных | Размер | Диапазон значений |
| Клиент | целочисленный | 4 байта | Уникальный идентификатор, положительные целые числа |
| Скидка | число двойной точности | 8 байт | Положительное число |

Структура сущности «Штрафы» описывается в [Таблице 6.](#таблица6)

[Таблица 6 - Структура сущности «Штрафы»](#таблица6текст)

|  |  |  |  |
| --- | --- | --- | --- |
| Название параметра | Тип данных | Размер | Диапазон значений |
| Автомобиль | целочисленный | 4 байта | Положительные целые числа |
| Клиент | целочисленный | 4 байта | Положительные целые числа |
| Причина | символьный | 255 байт | Текстовые символы |
| Размер штрафа | целочисленный | 4 байта | Положительные целые числа |
| Штраф | целочисленный | 4 байта | Положительные целые числа |
| Заказ | целочисленный | 4 байта | Положительные целые числа |

Диаграмма в нотации Питера Чена представлена на [Рисунке 1.](#рисунок1)

Изображение выглядит как диаграмма, зарисовка, Технический чертеж, линия

Автоматически созданное описание

[Рисунок 1 - Диаграмма в нотации Питера Чена](#рисунок1текст)

## **2.2 Логическое проектирование базы данных**

При логическом проектировании базы данных на основе реляционной модели данных, необходимо определить структуру базы данных и связи между таблицами, а также установить первичные и внешние ключи.

Для предметной области «Прокат автомобилей», на основе сущностей, представленных ранее, можно определить следующие таблицы:

* Автомобили (код автомобиля, марка автомобиля, стоимость автомобиля, стоимость аренды, тип автомобиля, код заказа, дата выхода, полное название, количество);
* Аренды (код автомобиля, код клиента, дата выдачи, дата возврата, счет, код заказа, состояние автомобиля, реальная дата возврата);
* Клиенты (код клиента, фамилия, имя, отчество, адрес, номер телефона);
* Постоянные клиенты (код клиента, скидка);
* Штрафы (код автомобиля, код клиента, причина, размер штрафа, код штрафа, код заказа);

Связи между таблицами можно определить следующим образом:

Связь один ко многим между таблицами Автомобили и Аренды по полю Код автомобиля.

Связь многие ко многим между таблицами Автомобили и Клиенты через промежуточную таблицу Аренды по полям Код автомобиля и Код клиента.

Связь один ко многим между таблицами Аренды и Штрафы по полям Код заказа, Код клиента. Код автомобиля.

Связь один к одному между таблицами Клиенты и Постоянные клиенты по полю Код клиента.

Объяснение связей:

Связь один ко многим между таблицами Автомобили и Аренды значит, что каждый автомобиль может быть арендован несколькими клиентами, но каждый наем автомобиля привязан только к одному автомобилю.

Связь многие ко многим между таблицами Автомобили и Клиенты через промежуточную таблицу Аренды значит, что каждый автомобиль может быть арендован несколькими клиентами, а каждый клиент может арендовать несколько автомобилей.

Связь один ко многим между таблицами Аренды и Штрафы значит, что каждая аренда автомобиля может привести к нескольким штрафам, но каждый штраф привязан только к одной аренде автомобиля.

Связь один к одному между таблицами Клиенты и Постоянные клиенты значит, что одному клиенту присуща одна запись в таблице Постоянные клиенты

На основе этих данных можно сформировать список первичных и внешних ключей.

Первичные ключи:

* В таблице Автомобили: Код автомобиля
* В таблице Клиенты: Код клиента
* В таблице Постоянные клиенты: Код клиента
* В таблице Аренды: Код заказа
* В таблице Штрафы: Код штрафа

Внешние ключи:

* В таблице Аренды: Код автомобиля, ссылается на поле Код автомобиля таблицы Автомобили; Код клиента, ссылается на поле Код клиента таблицы Клиенты
* В таблице Штрафы: Код автомобиля, ссылается на поле Код автомобиля таблицы Аренды; Код клиента, ссылается на поле Код клиента таблицы Аренды; Код заказа, ссылается на поле Код заказа таблицы Аренды
* В таблице Постоянные клиенты: Код клиента, ссылается на поле Код клиента таблицы Клиенты

ER-диаграмма с указанием сущностей, их атрибутов и отношений представлена на [Рисунке 2.](#рисунок2)

Изображение выглядит как текст, диаграмма, Параллельный, линия

Автоматически созданное описание

[Рисунок 2 - ER-диаграмма](#рисунок2текст)

## **2.3 Выбор и описание СУБД**

Выбор системы управления базами данных (СУБД) – это важный этап при проектировании базы данных. На рынке представлено множество СУБД разных типов, открытых и коммерческих, реляционных и нереляционных, облачных и локальных.

Для базы данных предметной области "Прокат автомобилей» была выбрана реляционная СУБД PostgreSQL.

PostgreSQL — это объектно-реляционная система управления базами данных (ORDBMS), которая поддерживает расширяемую архитектуру и обеспечивает высокую производительность и надежность.

Преимущества выбора PostgreSQL для данной базы данных:

* Open-source: PostgreSQL доступен бесплатно и имеет открытый исходный код. Это означает, что вы можете скачать, использовать и изменять PostgreSQL, не платя за это ни копейки.
* Надежность: PostgreSQL обеспечивает высокую надежность и устойчивость благодаря своим уникальным механизмам контроля целостности данных и механизмам восстановления после сбоев.
* Расширяемость: PostgreSQL имеет модульную архитектуру, которая позволяет пользователю легко расширять функциональность базы данных, добавляя новые возможности и типы данных.
* Поддержка SQL: PostgreSQL поддерживает стандарты SQL и ANSI SQL, что делает его более привлекательным для разработчиков и аналитиков, которые используют стандартный язык для извлечения данных.
* Полноценное хранилище данных: PostgreSQL обеспечивает многопоточный доступ и полноценную надежность хранения данных, что делает его идеальным выбором для крупных проектов.
* Скалируемость: PostgreSQL обладает возможностью масштабирования горизонтально и вертикально. Вы можете использовать механизмы репликации, а также использовать облачные сервисы для управления масштабированием PostgreSQL.
* Совместимость с операционными системами: PostgreSQL поддерживает большинство операционных систем, включая Windows, Linux, FreeBSD, macOS и другие, что делает его удобным в использовании на любой платформе.
* Большое сообщество разработчиков: PostgreSQL имеет огромное и активное сообщество разработчиков, которые постоянно обновляют и улучшают продукт, а также предоставляют множество документации, обучающих программ и примеров.

В целом, PostgreSQL — это мощная и надежная система управления базами данных для любой организации, она выделяется своей скоростью, надёжностью и гибкостью, что делает ее привлекательной для больших и малых проектов.

Основные преимущества PostgreSQL:

* поддержка транзакций и целостности данных;
* высокая скорость обработки данных;
* поддержка больших объемов данных;
* многопоточность и масштабируемость;
* наличие механизмов репликации и резервного копирования данных.

PostgreSQL поддерживает SQL-стандарт, что облегчает работу с базой данных и упрощает перенос приложений между разными СУБД. Она также поддерживает многие дополнительные функции, такие как JSON-обработка, полнотекстовый поиск и многое другое.

Одним из преимуществ PostgreSQL является его открытый исходный код, что позволяет пользователям изменять и дополнять его функциональность по своему усмотрению. Кроме того, это позволяет сократить расходы на приобретение и обновление программного обеспечения.

В целом, PostgreSQL является надежной, функциональной и расширяемой СУБД, которая подходит для создания баз данных различных масштабов и сложности.

## **2.4 Физическое проектирование базы данных**

Для физического проектирования базы данных предметной области «Прокат автомобилей» с использованием СУБД PostgreSQL, мы создадим базу данных, таблицы, функции и связи на основе реляционной модели данных, которую мы разработали ранее.

Для начала создадим базу данных с помощью SQL запроса

CREATE DATABASE carsharing

Далее уже будем создавать таблицы для созданной базы данных

1. Создание таблицы «Автомобили» представлено на Рисунке 3:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 3 - Создание таблицы «Автомобили»

1. Создание таблицы «Аренды» представлено на Рисунке 4:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 4 - Создание таблицы «Аренды»

1. Создание таблицы «Клиенты» представлено на Рисунке 5:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 5 - Создание таблицы «Клиенты»

1. Создание таблицы «Постоянные клиенты» представлено на Рисунке 6:

Изображение выглядит как текст, Шрифт, линия, число

Автоматически созданное описание

Рисунок 6 - Создание таблицы «Постоянные клиенты»

1. Создание таблицы «Штрафы» представлено на Рисунок 7 - Создание таблицы «Штрафы»:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 7 - Создание таблицы «Штрафы»

После создания таблиц можно приступать к написанию функций и триггеров:

1. Функция для расчета среднего чека для каждого клиента с группировкой по убыванию значений чека представлена на Рисунке 8:

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 8 - Создание функции для расчета среднего чека

1. Функция для расчета минимального количества месяцев, которое потребуется для того, чтобы погасить каждый из штрафов представлена на Рисунке 9:

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

Рисунок 9 - Создание функции для расчета минимального количества месяцев, которое потребуется для того, чтобы погасить каждый из штрафов

1. Функция для расчета среднего интервала времени между заказами выбранного клиента представлена на Рисунке 10:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 10 – Создание функции для расчета среднего интервала времени между заказами выбранного клиента

1. Триггер, который: При добавлении записи в таблицу заказов рассчитывает и устанавливает предварительную стоимость услуги(стоимость рассчитывается исходя из даты выпуска автомобиля и ее стоимости, а также от количества дней аренды) представлен на Рисунке 11:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 11 - Создание триггера расчета и обновления стоимости

1. Триггер, который изменяет количество автомобилей в наличии при вставке строки в таблицу inuse, при обновлении же, если состояние автомобиля не расценено, как 1, то прибавляет возвращенное транспортное средство к вышеупомянутому количеству представлен на Рисунке 12:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 12 - Создание триггера для регулировки значений количества автомобилей в автопарке

1. Триггер, который, при обновлении записей:

1) Проверяет в назначенную ли дату был возвращен автомобиль и если нет, добавляет клиента в таблицу со штрафами и назначает штраф

2) Проверяет состояние машины, в случае если состояние машины меньше пяти. Назначает штраф и добавляет клиента в таблицу со штрафами. В случае если клиент полностью вывел транспортное средство из эксплуатации, ему назначается штраф в размере 1.3 от стоимости транспортного средства представлен на Рисунке 13:

Изображение выглядит как текст, снимок экрана, Шрифт, документ

Автоматически созданное описание

Рисунок 13 - Создание триггера для назначения клиентам штрафов

1. Триггер, который высчитывает количество заказов клиента, и в случае, если количество заказов превышает число 3 назначает клиенту скидку в размере, зависящем от количества заказов:

* 3 заказа - Коэффициент=0.95 (5%)
* 6 заказов - Коэффициент=0.9 (10%)
* 10 заказов - Коэффициент=0.8 (20%)

Представлен на Рисунке 14

Изображение выглядит как текст, снимок экрана, Шрифт, документ

Автоматически созданное описание

Рисунок 14 - Создание триггера для присвоения клиенту статуса постоянного клиента

1. Триггер, который при обновлении таблицы inuse, в случае если клиент вернул автомобиль в состоянии 1 (вывод транспортного средства из строя) запрещает клиенту делать последующие заказы, выводя сообщение о том, что клиент находится в черном списке, представлен на рисунке 15.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 15 - Создание триггера на таблицу «Аренды», который добавляет клиента в «черный список», запрещая ему делать последующие заказы

Итого было создано 5 таблиц, 3 функции, 5 триггеров.

# **3 Использование базы данных**

## **3.1 Создание DataSet**

Таблица 7 - DataSet для таблицы «Клиенты»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Код клиента | Имя | Фамилия | Отчество | Адрес | Номер телефона |
| 1 | Кирилл | Никонов | Игоревич | ул. Часовая 13 | 89178549933 |
| 2 | Андрей | Никонов | Игоревич | ул. Семёновская 43 | 89278949533 |
| 3 | Сергей | Алексеев | Семенович | ул. Спартаковская 1 | 89296494477 |
| 4 | Валерия | Вишневская | Олеговна | ул. Песчаная 3с3 | 89510648232 |
| 5 | Олеся | Васильева | Михайловна | ул. Соколова 5 | 89138420055 |
| 6 | Виталий | Михайлов | Тимурович | ул. Тимирязевская 12 | 89192731363 |

Таблица 8 - DataSet для таблицы «Аренды»

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Код автомобиля | Код клиента | Дата выдачи | Дата возврата | Код заказа | Реальная дата возврата | Состояние автомобиля | Код заказа |
| 27/07/2021 | 5 | 1 | 27/07/2021 | 1 | 27/07/2021 | 5 | 1 |
| 29/07/2021 | 5 | 2 | 29/07/2021 | 2 | 29/07/2021 | 5 | 2 |
| 05/09/2021 | 5 | 3 | 05/09/2021 | 3 | 05/09/2021 | 5 | 3 |
| 02/05/2019 | 5 | 4 | 02/05/2019 | 4 | 02/05/2019 | 5 | 4 |
| 07/06/2019 | 4 | 5 | 07/06/2019 | 5 | 07/06/2019 | 4 | 5 |
| 05/02/2022 | 4 | 6 | 05/02/2022 | 6 | 05/02/2022 | 4 | 6 |
| 08/09/2021 | 5 | 7 | 08/09/2021 | 7 | 08/09/2021 | 5 | 7 |
| 17/09/2021 | 5 | 8 | 17/09/2021 | 8 | 17/09/2021 | 5 | 8 |
| 24/09/2021 | 5 | 9 | 24/09/2021 | 9 | 24/09/2021 | 5 | 9 |
| 29/10/2021 | 5 | 10 | 29/10/2021 | 10 | 29/10/2021 | 5 | 10 |
| 31/10/2021 | 5 | 11 | 29/10/2021 | 11 | 31/10/2021 | 5 | 11 |

Таблица 9 - DataSet для таблицы «Транспортные средства»

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Код автомобиля | Марка автомобиля | Цена автомобиля | Цена проката/день | Тип | Дата выхода | Модель | Количество |
| 1 | TOYOTA | 2500000 | 14500 | купе | 2021-11-17 | TOYOTA GR 86 | 20 |
| 2 | TOYOTA | 1700000 | 11500 | универсал | 2015-07-16 | TOYOTA FORTUNER | 30 |
| 3 | TOYOTA | 2800000 | 17500 | универсал | 2021-05-17 | TOYOTA YARIS CROSS | 25 |
| 4 | FORD | 2900000 | 21500 | универсал | 2019-01-09 | FORD EXPLORER | 15 |
| 5 | MERCEDES | 4900000 | 45000 | универсал | 2021-07-09 | MERCEDES-BENZ GLS MAYBACH | 10 |
| 6 | FORD | 2900000 | 21500 | хэтчбек | 2016-07-09 | FORD KA+ | 20 |

## **3.2 Добавление данных в базу**

Команда INSERT в PostgreSQL используется для добавления новых строк (записей) в таблицу базы данных. Она позволяет указывать значения для каждого столбца таблицы, которые будут сохранены в новой строке. Формат команды INSERT выглядит следующим образом:

INSERT INTO table\_name (column1, column2, column3, ...)

VALUES (value1, value2, value3, ...);

Команда UPDATE в PostgreSQL используется для обновления существующих строк (записей) в таблице базы данных. Она позволяет изменять значения в указанных столбцах таблицы для выбранных строк. Формат команды UPDATE выглядит следующим образом:

UPDATE table\_name

SET column1 = value1, column2 = value2, ...

WHERE condition;

Для заполнения таблиц мы как раз таки и будем использовать эти команды

Приведем примеры добавления данных в каждую из таблиц.

1. Добавление данных в таблицу «Аренды» представлено на Рисунке 16:

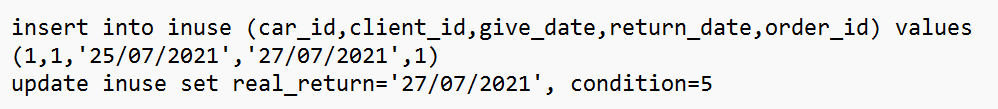


Рисунок 16 - Добавление данных в таблицу «Аренды»

1. Добавление данных в таблицу «Автомобили» представлено на Рисунке 17:

Изображение выглядит как текст, Шрифт, снимок экрана, линия

Автоматически созданное описание

Рисунок 17 - Добавление данных в таблицу «Автомобили»

1. Добавление данных в таблицу «Клиенты» представлено на Рисунке 18:

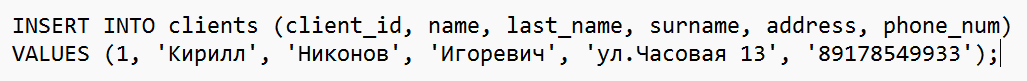


Рисунок 18 - Добавление данных в таблицу «Клиенты»

## **Запросы к базе данных**

Запрос (query) — это команда на языке SQL, которая используется для выполнения операций с данными в базе данных. Запросы могут быть использованы для добавления, изменения или удаления данных, а также для выборки нужных данных из таблицы.

Запросы SQL — это основной способ взаимодействия с базами данных. Они позволяют пользователю получать ответы на свои запросы и извлекать информацию из базы данных, которую он хочет получить. В запросах SQL можно использовать различные условия, операторы и функции для выборки определенных данных, фильтрации данных, объединения таблиц и многого другого.

В данном параграфе будут представлены тематических 20 запросов с использованием языка SQL.

1. Запрос, который выводит фамилии клиентов, которые потратили на прокат и штрафы сумму в общем превышающую 200000 представлен на Рисунке 19:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание

Рисунок 19 - Запрос для получения списка фамилий клиентов, потративших в общей сумме более 200000

1. Запрос для вывода клиента, которым было арендовано наибольшее количество автомобилей, учитывая то, что в таблице присутствуют однофамильцы представлен на Рисунке 20:

Изображение выглядит как текст, электроника, снимок экрана, дисплей

Автоматически созданное описание

Рисунок 20 - Запрос для вывода клиента, арендовавшего наибольшее количество автомобилей

1. Запрос, который вычисляет процентное изменение прибыли автопарка по годам представлен на Рисунке 21:

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок 21 – Запрос для вычисления процентного изменения прибыли автопарка по годам

1. Запрос для вывода номеров заказов по каждому дню недели представлен на Рисунке 22:

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 22 - Запрос для вывода дней недели и заказов, которые были совершены в данные дни

1. Запрос для вывода клиентов и их номеров, общая сумма штрафов которых превышает 10 процентов от суммарной стоимости автомобилей (по 1 единице транспортного средства) представлен на Рисунке 23:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание

Рисунок 23 - Запрос для получения списка клиентов, общая сумма штрафов которых превышает 10% от стоимости всех автомобилей (по 1 единице)

1. Запрос для вывода информации о заданной модели автомобиля представлен на Рисунке 24:

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 24 - Запрос для вывода информации о заданной модели автомобиля

1. Запрос для вывода информации о деятельности клиентов представлен на Рисунке 25:

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 25 - Запрос для вывода информации о деятельности клиентов автопарка

1. Запрос для вывода общих затрат на автопарк с момента первой аренды представлен на Рисунке 26:

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рисунок 26 - Запрос для вывода общих затрат на автопарк с момента первой аренды

1. Запрос для поиска всех автомобилей, которые не были в использовании последние 3 месяца представлен на Рисунке 27:

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рисунок 27 - Запрос для поиска всех автомобилей, которые не были в использовании последние 3 месяца

1. Запрос для вывода всех клиентов, арендовавших автомобили, которые попадали в аварию, а также количество их аренд, представлен на Рисунке 28:

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рисунок 28 - Запрос для вывода всех клиентов, арендовавших автомобили, которые попадали в аварию, и их количество аренд

1. Запрос для вывода списка всех клиентов с заданным форматированием представлен на Рисунке 29:

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 29 – Запрос для вывода списка всех клиентов с заданным форматированием

1. Запрос для вывода количества автомобилей на данный момент, количества автомобилей, полностью выведенных из строя и исходного количества автомобилей представлен на Рисунке 30:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание

Рисунок 30 - Запрос для вывода количества автомобилей на данный момент, количества автомобилей, полностью выведенных из строя и исходного количества автомобилей

1. Запрос для распределения клиентов на VIP и NON-VIP в зависимости от потраченной на прокаты суммы представлен на Рисунке 31:

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 31 - Запрос для распределения клиентов на VIP и NON-VIP

1. Запрос для подсчета процентов для каждого типа штрафов от общего количества записей, а также вывода общей суммы назначенных выплат для них представлен Рисунке 32:

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рисунок 32 - Запрос для подсчета процентов для каждого типа штрафов от общего количества записей, а также вывода общей суммы назначенных выплат для каждого типа штрафов

1. Запрос для вывода процента «надежности» клиента по заданным параметрам представлен на Рисунке 33:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 33 - Запрос для вывода процента «надежности» клиента

1. Запрос для вывода списка информации по автомобилям по заданным характеристикам представлен на Рисунке 34:

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рисунок 34 - Запрос для вывода списка информации по автомобилям по заданным характеристикам

1. Запрос для вывода измененной стоимости в виде дополнительной скидки для заказов, сделанных в четный, нечетный месяц и в 29 число любого месяца представлен на Рисунке 35:

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 35 - Запрос для вывода измененной стоимости для заказов, сделанных в четный, нечетный месяц и в 29 число любого месяца

1. Запрос для назначения характеристики штрафа по заданным параметрам представлен на Рисунке 36:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 36 - Запрос для назначения характеристики штрафа по заданным параметрам

1. Запрос для вывода информации по хотя бы раз арендованным моделям автомобилей, представлен на Рисунке 37:

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 37 - Запрос для вывода информации по хотя бы раз арендованным моделям автомобилей

1. Запрос для расчета и вывода разницы (в процентах) между общими суммами всех заказов двух клиентов (при этом клиенты обязательно должны быть однофамильцами) представлен на Рисунке 38:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание

Рисунок 38 - Запрос для расчета и вывода разницы (в процентах) между общими суммами всех заказов у двух однофамильцев

## **3.4 Тестирование функций и триггеров**

Тестирование функций и триггеров в базах данных — это процесс проверки работоспособности и правильности функционирования процедур, которые предназначены для автоматизации определенных задач в базе данных. Функции и триггеры могут выполнять различные действия, такие как проверка условий, обеспечение целостности данных, автоматическое обновление информации и т. д.

Цель тестирования функций и триггеров — это убедиться, что они работают правильно и не приводят к ошибкам в базе данных. Тестирование помогает идентифицировать и устранить проблемы, связанные с функционированием процедур, которые могут привести к нарушению целостности данных или ошибкам в работе приложений, использующих данную базу данных.

Подробный процесс тестирования функций и триггеров может включать в себя следующие шаги:

* Подготовительный этап: здесь определяются функции и триггеры, которые необходимо протестировать.
* Составление тестовых сценариев: здесь формулируются конкретные тестовые сценарии, которые проверяют функциональность и правильность работы каждой функции и триггера. На этом этапе определяются ожидаемые результаты и действия, которые должны быть выполнены.
* Создание тестовых данных: здесь создаются тестовые данные для запуска тестовых сценариев. Это могут быть фиктивные данные или копии реальных данных в базе данных.
* Запуск тестовых сценариев: на этом этапе запускаются определенные тестовые сценарии для проверки функций и триггеров. Результаты тестирования фиксируются.
* Анализ результатов тестирования: на этом этапе производится анализ результатов тестирования, для определения наличия ошибок и аномалий в работе функций и триггеров. Если обнаружены ошибки, то их необходимо исправить и повторить этапы 4 и 5 до тех пор, пока все ошибки не будут устранены.
* Документирование результатов: результаты тестирования и исправления ошибок должны быть документированы для последующего анализа и мониторинга функций и триггеров.

Тестовые случаи для функции avg\_bill\_per\_client ():

1. Проверка, что функция корректно вычисляет средний счет для каждого клиента:

* Создать несколько клиентов и соответствующих заказов в таблице inuse с разными суммами счетов.
* Запустить функцию avg\_bill\_per\_client ().
* Проверить, что функция возвращает корректные значения среднего счета для каждого клиента.

Результат теста приведен на Рисунке 39.

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рисунок 39 – Тест (1) функции avg\_bill\_per\_client ()

1. Проверка сортировки возвращаемых данных:

* Создать несколько клиентов и соответствующих заказов в таблице inuse с разными суммами счетов.
* Запустить функцию avg\_bill\_per\_client().
* Проверить, что функция возвращает значения в правильном порядке, от самого большого среднего счета к самому маленькому.

1. Проверка корректности обработки пустой таблицы inuse:

* Очистить таблицу inuse.
* Запустить функцию avg\_bill\_per\_client().
* Проверить, что функция не возвращает никаких значений.
* Проверка корректности обработки пустой таблицы clients:
* Очистить таблицу clients.
* Запустить функцию avg\_bill\_per\_client().
* Проверить, что функция не возвращает никаких значений.

1. Проверка корректности формата возвращаемых значений:

* Создать клиента и соответствующий заказ в таблице inuse с суммой счета.
* Запустить функцию avg\_bill\_per\_client().
* Проверить, что возвращаемые значения соответствуют заданному формату (client\_id integer, first\_name varchar(50), last\_name varchar(50), avg\_bill numeric(10,2)).

Тестовые случаи для функции calculate\_months\_to\_pay (month\_pay):

1. Проверка корректности вычисления количества месяцев на выплату штрафа:

* Создать несколько штрафов с разными суммами в таблице fines.
* Запустить функцию calculate\_months\_to\_pay() с выбранным значением monthly\_payment.
* Проверить, что функция корректно вычисляет количество месяцев на выплату каждого штрафа по формуле CEIL(cost / monthly\_payment).

Результат теста приведен на Рисунке 40.

Изображение выглядит как текст, снимок экрана, диаграмма, число

Автоматически созданное описание

Рисунок 40 – Тест (1) функции calculate\_months\_to\_pay(month\_pay)

1. Проверка корректности обработки штрафов с нулевой стоимостью:

* Создать штраф с нулевой стоимостью в таблице fines.
* Запустить функцию calculate\_months\_to\_pay() с любым значением monthly\_payment.
* Проверить, что функция возвращает значение months\_to\_pay равным 0 для этого штрафа.

Тестовые случаи для функции calculate\_avg\_time\_between\_orders(client):

1. Проверка корректности обработки клиентов с одним заказом:

* Создать заказ для клиента в таблице inuse.
* Запустить функцию calculate\_avg\_time\_between\_orders() для данного клиента.
* Проверить, что функция возвращает 0, так как между заказами нет интервалов.

Результат теста приведен на Рисунке 41.

Изображение выглядит как текст, программное обеспечение, снимок экрана

Автоматически созданное описание

Рисунок 41 – Тест (1) функции calculate\_avg\_time\_between\_orders(client\_id)

1. Проверка корректности вычисления среднего интервала между заказами клиента:

* Создать несколько заказов для клиента с разными датами выдачи и возврата в таблице inuse.
* Запустить функцию calculate\_avg\_time\_between\_orders() для данного клиента.
* Проверить, что функция корректно вычисляет средний интервал между заказами по формуле (сумма всех интервалов между заказами) / (количество заказов - 1).

Результат теста приведен на Рисунке 42.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 42 – Тест (2) функции calculate\_avg\_time\_between\_orders(client\_id)

1. Проверка корректности обработки клиентов без заказов:

* Удалить из таблицы inuse все заказы для данного клиента.
* Запустить функцию calculate\_avg\_time\_between\_orders() для данного клиента.
* Проверить, что функция возвращает 0, так как заказов нет.

1. Проверка корректности формата возвращаемых значений:

* Создать несколько заказов для клиента в таблице inuse.
* Запустить функцию calculate\_avg\_time\_between\_orders() для данного клиента.
* Проверить, что возвращаемое значение соответствует заданному формату INTERVAL.

Тестовые случаи для триггера trigger\_bill():

1. Проверка корректности вычисления стоимости аренды автомобиля:

* Создать заказ для клиента с автомобилем, выпущенным в разные годы, в таблице inuse.
* Запустить триггер trigger\_bill() для каждого заказа.
* Проверить, что триггер корректно вычисляет стоимость аренды автомобиля по формуле:
  + для автомобилей, выпущенных между 2000 и 2005 годами: стоимость = share\_cost \* 1.5 \* (количество дней аренды) \* (скидка клиента);
  + для автомобилей, выпущенных между 2005 и 2015 годами: стоимость = share\_cost \* 2.5 \* (количество дней аренды) \* (скидка клиента);
  + для автомобилей, выпущенных между 2015 и 2023 годами: стоимость = share\_cost \* 3.5 \* (количество дней аренды) \* (скидка клиента).

Результат теста приведен на Рисунке 43.

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 43 – Тест (1) триггера trigger\_bill

1. Проверка корректности обработки заказа с клиентом без скидки:

* Создать заказ для клиента без скидки в таблице inuse.
* Запустить триггер trigger\_bill() для данного заказа.
* Проверить, что триггер корректно вычисляет стоимость аренды автомобиля без учета скидки клиента.

1. Проверка корректности обработки удаления заказа:

* Создать заказ для клиента в таблице inuse.
* Запустить триггер trigger\_bill() для данного заказа.
* Удалить данный заказ из таблицы inuse.
* Проверить, что триггер корректно удаляет данные стоимости аренды автомобиля из таблицы inuse.

1. Проверка корректности формата обновляемых значений:

* Создать заказ для клиента с автомобилем в таблице inuse.
* Запустить триггер trigger\_bill() для данного заказа.
* Проверить, что обновляемые значения стоимости аренды автомобиля в таблице inuse соответствуют заданному формату числа с плавающей запятой.

Тестовые случаи для триггера adjust\_car\_amount\_trigger():

1. Проверка корректности обновления числа автомобилей после заказа:

* Создать заказ для автомобиля в таблице inuse.
* Обновить информацию о заказе.
* Проверить, что триггер adjust\_car\_amount\_trigger() корректно обновляет количество доступных автомобилей при создании и изменении заказа.

Результат теста приведен на Рисунке 44.

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Автоматически созданное описание

Рисунок 44 – Тест (1) триггера adjust\_car\_amount\_trigger

1. Проверка корректности обработки возврата автомобиля:

* Создать заказ для автомобиля в таблице inuse.
* Возвратить автомобиль.
* Проверить, что триггер adjust\_car\_amount\_trigger() корректно обновляет количество доступных автомобилей при возврате автомобиля.

Результат теста приведен на Рисунке 45.

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рисунок 45 – Тест (2) триггера adjust\_car\_amount\_trigger

1. Проверка корректности обработки изменения состояния автомобиля:

* Создать заказ для автомобиля в таблице inuse.
* Обновить информацию о состоянии автомобиля.
* Проверить, что триггер adjust\_car\_amount\_trigger() не обновляет количество доступных автомобилей при изменении состояния автомобиля.

1. Проверка корректности формата обновляемых значений:

* Создать заказ для автомобиля в таблице inuse.
* Обновить информацию о заказе и/или автомобиле.
* Проверить, что обновляемые значения количества доступных автомобилей в таблице cars соответствуют заданному формату целого числа.

Тестовые случаи для триггера process\_fines\_trigger() :

1. Проверка корректности создания штрафа за незначительные повреждения:

* Создать заказ для автомобиля в таблице inuse
* Установить состояние в "Незначительные повреждения"
* Завершить заказ, указав дату возврата
* Проверить, что триггер process\_fines\_trigger() корректно создает штраф с правильной стоимостью и описанием.

Результат теста приведен на Рисунке 46.

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Автоматически созданное описание

Рисунок 46 – Тест (1) триггера process\_fines\_trigger

1. Проверка корректности создания штрафа за значительные повреждения:

* Создать заказ для автомобиля в таблице inuse
* Установить состояние в "Значительные повреждения"
* Завершить заказ, указав дату возврата
* Проверить, что триггер process\_fines\_trigger () корректно создает штраф с правильной стоимостью и описанием;

1. Проверка корректности создания штрафа за вывод автомобиля из эксплуатации

* Создать заказ для автомобиля в таблице inuse
* Установить состояние в "Вывод из эксплуатации"
* Завершить заказ, указав дату возврата
* Проверить, что триггер process\_fines\_trigger () корректно создает штраф с правильной стоимостью и описанием;

1. Проверка корректности создания штрафа за просрочку даты возврата:

* Создать заказ для автомобиля в таблице inuse
* Не завершать его
* Через несколько дней (больше, чем указанное при создании заказа время аренды) указать фактическую дату возврата и завершить заказ
* Проверить, что триггер process\_fines\_trigger() корректно создает штраф с правильной стоимостью и описанием;

Тестовые случаи для триггера check\_discount:

1. Проверка корректности добавления клиента в список постоянных при количестве заказов > = 3 в любом году:

* Создать три заказа для одного и того же клиента в определенном году;
* Убедиться, что клиент появился в таблице постоянных, запустив запрос SELECT \* FROM val\_clients

Результат теста приведен на Рисунке 47.

Изображение выглядит как текст, снимок экрана, линия, Шрифт

Автоматически созданное описание

Рисунок 47 – Тест (1) триггера check\_discount()

1. Проверка корректности добавления клиента в список постоянных только при первом достижении количества заказов >= 3:

* Создать два заказа для одного и того же клиента в разные годы (например, 2019, 2020);
* Убедиться, что клиент не находится в таблице постоянных, запустив запрос SELECT \* FROM val\_clients WHERE client\_id = <id\_клиента>;
* Создать еще один заказ для этого клиента в 2021 году;
* Убедиться, что клиент добавился в таблицу постоянных с коэффициентом скидки 0.95, запустив запрос SELECT \* FROM val\_clients WHERE client\_id = <id\_клиента>;
* Создать еще один заказ для этого клиента в 2022 году;
* Убедиться, что клиент остался в таблице постоянных с коэффициентом скидки 0.95, запустив запрос SELECT \* FROM val\_clients WHERE client\_id = <id\_клиента>;

1. Проверка корректности отсутствия добавления клиента в список постоянных при количестве заказов < 3:

* Создать два заказа для одного и того же клиента в один год (например, 2019);
* Убедиться, что клиент не находится в таблице постоянных, запустив запрос SELECT \* FROM val\_clients WHERE client\_id = <id\_клиента>;
* Создать еще один заказ для этого клиента в 2020 году;
* Убедиться, что клиент не добавился в таблицу постоянных, запустив запрос SELECT \* FROM val\_clients WHERE client\_id = <id\_клиента>;

1. Проверка корректности отсутствия добавления клиента в список постоянных, если он уже есть в таблице штрафов:

* Создать три заказа для одного и того же клиента в разные годы (например, 2019, 2020, 2021);
* Создать штраф для этого клиента;
* Убедиться, что клиент не добавился в таблицу постоянных, запустив запрос SELECT \* FROM val\_clients WHERE client\_id = <id\_клиента>;

1. Проверка корректности обновления коэффициента скидки при добавлении клиента в таблицу постоянных:

* Создать два заказа для одного и того же клиента в разные годы (например, 2019, 2020);
* Убедиться, что клиент не находится в таблице постоянных, запустив запрос SELECT \* FROM val\_clients WHERE client\_id = <id\_клиента>;
* Создать еще один заказ для этого клиента в 2021 году;
* Убедиться, что клиент добавился в таблицу постоянных с коэффициентом скидки 0.95, запустив запрос SELECT \* FROM val\_clients WHERE client\_id = <id\_клиента>;
* Создать еще один заказ для этого клиента в 2022 году;
* Убедиться, что клиент остался в таблице постоянных с обновленным коэффициентом скидки (0.9 или меньше), запустив запрос SELECT \* FROM val\_clients WHERE client\_id = <id\_клиента>;

Тестовые случаи для триггера check\_blacklist() :

1. Проверка корректности блокировки добавления записи при наличии клиента в таблице штрафов:

* Создать запись в таблице штрафов для определенного клиента с причиной 'Вывод средства из эксплуатации';
* Попытаться создать новую запись в таблице inuse для этого же клиента;
* Убедиться, что возникает ошибка "Вы находитесь в черном списке".

Результат теста приведен на Рисунке 48.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 48 – Тест (1) триггера check\_blacklist

1. Проверка корректности добавления записи при отсутствии клиента в таблице штрафов:

* Создать нового клиента и добавить для него запись в таблицу inuse;
* Убедиться, что запись добавилась без ошибок.

1. Проверка корректности работы триггера на обновление записи в таблице:

* Создать нового клиента и добавить для него запись в таблицу inuse;
* Добавить штраф для этого клиента с причиной 'Вывод средства из эксплуатации';
* Попытаться обновить информацию о клиенте в таблице inuse;
* Убедиться, что возникает ошибка "Вы находитесь в черном списке" и обновление не происходит.

1. Проверка корректности работы триггера при удалении записи из таблицы:

* Создать нового клиента и добавить для него запись в таблицу inuse;
* Добавить штраф для этого клиента с причиной 'Вывод средства из эксплуатации';
* Попытаться удалить запись о клиенте из таблицы inuse;
* Убедиться, что возникает ошибка "Вы находитесь в черном списке" и удаление не происходит.

# **ЗАКЛЮЧЕНИЕ**

Как результат проведенной работы по разработке базы данных для проката автомобилей в СУБД PostgreSQL были достигнуты важные запланированные цели:

* Были закреплены теоретические знания и практические навыки, связанные с созданием эффективных баз данных в PostgreSQL, благодаря основательному изучению специальной литературы по этой теме.
* Была успешно разработана база данных, которая включает в себя сущности, такие, как автомобили, аренды, клиенты, постоянные клиенты и штрафы, а также обеспечивает целостность данных и поддержку связей между ними в предметной области проката автомобилей.

Кроме того, в процессе работы были применены различные методы и инструменты, такие как проектирование модели данных, создание схемы базы данных, написание SQL-запросов и настройка конфигурации PostgreSQL. В результате, было получено более глубокое понимание принципов проектирования баз данных, а также обогащен практический опыт в работе с PostgreSQL.

Таким образом, результаты этой работы могут быть использованы для улучшения процессов проката автомобилей, а также для дальнейшего исследования и разработки в данной области.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. ER-модель. — Режим доступа: <https://ru.wikipedia.org/wiki/ER-модель> (дата обращения: 29.05.2023).
2. Определение связей между таблицами в базах данных. Режим доступа: [https://learn.microsoft.com/ru-ru/office/troubleshoot/access/define-table-relationships](https://learn.microsoft.com/ru-ru/office/troubleshoot/access/define-table-relationships%20) (дата обращения: 29.05.2023).
3. SQL | Oracle Help Center. — Режим доступа: <https://docs.oracle.com/en/database/oracle/oracle-database/19/cncpt/sql.html#GUID-B383E550-BBF9-4449-A6B0-56B3B3C81A09> (дата обращения: 29.05.2023).
4. Диаграммы «сущность-связь»: нотация Чена. — Режим доступа: <https://studfile.net/preview/5265875/page:18/> (дата обращения: 31.05.2023).
5. PostgreSQL: Documentation: 15: CREATE TRIGGER — Режим доступа: <https://www.postgresql.org/docs/current/sql-createtrigger.html> (дата обращения: 20.05.2023).
6. PostgreSQL Functions | PostgreSQL Tutorials and Solutions — Режим доступа: <https://www.postgresql.r2schools.com/postgresql-functions/> (дата обращения: 20.05.2023).
7. Data Set: Definition, Types, Examples & Public Data Sets. Режим доступа: <https://analystanswers.com/data-set-definition-types-examples-public-data-sets/> (дата обращения: 01.06.2023).
8. «PostgreSQL. Основы языка SQL» — Е. Моргунов. Режим доступа: <https://postgrespro.ru/education/books/sqlprimer> (дата обращения 29.05.2023).
9. Триггерные процедуры в PostgreSQL. Режим доступа: <https://postgrespro.ru/docs/postgresql/9.6/plpgsql-trigger> (дата обращения 20.05.2023)
10. ГОСТ 7-32 2017 Режим доступа: <https://www.rea.ru/ru/org/managements/orgnirupr/Documents/gost_7.32-2017.pdf> (дата обращения 31.05.2023)