Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Модели данных и системы управления базами данных

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

**СТАНЦИЯ ТЕХНИЧЕСКОГО ОБСЛУЖИВАНИЯ**

БГУИР КП 1-40 04 01 059 ПЗ

Студент: гр. 053502

Шаргородский И.С.

Руководитель: ассистент кафедры информатики Плиска В. С.

Минск 2023

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ

Сфера автосервиса в современном мире претерпевает существенные изменения, подчеркивая важность эффективной организации и полного контроля за всеми аспектами своей операционной деятельности. Отслеживание процессов технического обслуживания, управление состоянием оборудования, оценка квалификации персонала и ведение клиентской базы – все эти составляющие требуют инновационного и системного подхода для обеспечения безупречного функционирования современного автосервиса.

Сложившаяся конкурентная среда и повышенные ожидания клиентов вынуждают автосервисы не только следить за техническим обслуживанием автомобилей, но и создавать вокруг себя интегрированную экосистему управления, где каждая деталь вносит свой вклад в общую картину эффективности и качества обслуживания.

Цель настоящей курсовой работы – разработать и проектировать базу данных, которая не только отражает разнообразие аспектов деятельности автосервиса, но и предоставляет инструменты для оперативного и системного управления этими аспектами. В рамках работы будут рассмотрены ключевые сущности, такие как техническое оборудование, квалификация персонала и клиентская база, а также предложен интегрированный подход к их управлению.

1. ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ И АНАЛИЗ АНАЛОГОВ
   1. Общие сведения предметной области

Автосервис, или станция технического обслуживания (СТО), представляет собой организацию, специализирующуюся на техническом обслуживании, диагностике и ремонте автотранспортных средств. Основная задача СТО заключается в обеспечении надежности, безопасности и эффективной эксплуатации автомобилей.

Техническое обслуживание включает в себя регулярные проверки, замену расходных материалов, а также ремонт неисправностей, выявленных в процессе диагностики. Специалисты СТО осуществляют работу с различным техническим оборудованием, используя разнообразные инструменты для обеспечения высокого стандарта обслуживания.

База данных (БД) в данной контексте представляет собой систему хранения и управления структурированной информацией, связанной с деятельностью автосервиса. Она включает в себя данные о клиентах, автомобилях, истории технического обслуживания, а также информацию о состоянии и ремонте технического оборудования. БД в автосервисе помогает эффективно отслеживать и управлять всеми аспектами бизнес-процессов, улучшая оперативность и качество предоставляемых услуг.

* 1. Назначение системы

Создание базы данных в контексте автосервиса (СТО) направлено на обеспечение эффективного управления информацией, необходимой для оперативного и качественного функционирования предприятия. Назначение системы охватывает несколько ключевых аспектов:

* Управление техническим оборудованием и ресурсами: База данных предоставляет инструменты для отслеживания состояния и обслуживания технического оборудования в автосервисе. Информация о статусе, технических характеристиках и истории обслуживания инструментов и машин позволяет эффективно планировать ресурсы и проводить своевременное обслуживание.
* Управление квалификацией и обучением персонала: БД включает в себя данные о квалификации сотрудников, их профессиональном развитии и обучении. Это позволяет легко выявлять потребности в обучении, контролировать уровень компетенции персонала и обеспечивать соответствие требованиям, предъявляемым к выполнению определенных видов работ.
* Ведение клиентской базы: Система хранит данные о клиентах, их автомобилях, истории посещений и предоставляемых услугах. Это позволяет автосервису управлять взаимоотношениями с клиентами, предоставлять персонализированный сервис и эффективно учитывать предпочтения клиентов.
* Оптимизация процессов и управление затратами: БД обеспечивает сбор и анализ данных, что позволяет выявлять узкие места в бизнес-процессах, оптимизировать использование ресурсов и управлять затратами. Это важный аспект для повышения эффективности и конкурентоспособности автосервиса.

Назначение создаваемой системы базы данных – обеспечение автосервиса современным инструментом для систематизации, управления и оптимизации всех аспектов его деятельности.

* 1. Анализ существующих аналогов

Перед разработкой базы данных для автосервиса важно проанализировать существующие аналоги, которые могут предоставлять подобные функциональности. Такой анализ позволяет выявить успешные решения, изучить их преимущества и недостатки, а также определить тенденции в данной области.

В качестве аналогичных программных продуктов рассматривались другие системы управления автосервисами или системы оптимизации.

1.3.1 Autofluent

Пример решения продемонстрирован на рисунке 1.1.

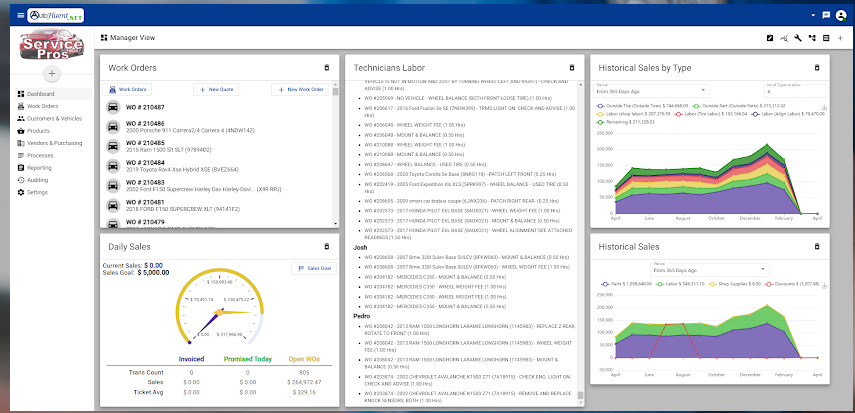


Рисунок 1.1 – Решение системы автоматизации Autofluent

Решение представляет собой интегрированную систему управления автосервисом, ориентированную на обеспечение эффективного ведения бизнеса в сфере технического обслуживания автомобилей. Рассмотрим ключевые характеристики данной платформы:

* Учет заказов и складской учет. Autofluent предоставляет всестороннюю систему учета заказов, позволяя легко отслеживать и управлять рабочим процессом. Интегрированный складской учет позволяет эффективно управлять запчастями и материалами.
* Мобильное приложение. Мобильное приложение Autofluent обеспечивает управление бизнесом из любой точки, что особенно важно в условиях подвижного рабочего процесса автосервисов. Это дает возможность оперативно реагировать на изменения и контролировать текущие заказы.
* Аналитика и отчетность. Система предоставляет возможность анализа и отчетности, что помогает более глубоко понимать бизнес-процессы и принимать обоснованные управленческие решения. Аналитические инструменты Autofluent могут помочь в выявлении тенденций и оптимизации стратегии.
* Недостатки. Autofluent может оказаться слишком затратным решением для небольших автосервисов, особенно в начальной стадии их развития.

В целом, Autofluent представляет собой мощное решение с широким функционалом для управления автосервисом. Однако стоит учитывать, что его высокая стоимость может ограничивать доступность для малых предприятий.

1.3.2 Maxpanda

Пример решения продемонстрирован на рисунке 1.1.

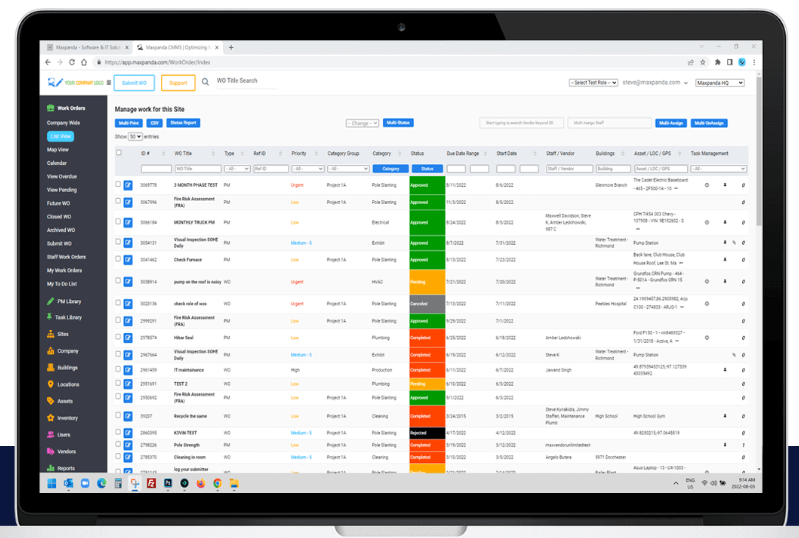


Рисунок 1.2 – Решение системы автоматизации Maxpanda

Maxpanda – это программное решение в сфере управления оборудованием и обслуживанием зданий, которое также находит применение в автосервисах. Рассмотрим ключевые особенности и характеристики этой системы:

* Простой интерфейс. Maxpanda известен своим простым и интуитивно понятным интерфейсом, что делает его привлекательным для пользователей с разным уровнем технической грамотности. Удобство использования способствует быстрой адаптации персонала.
* Управление оборудованием и материалами: Преимущества: Система предоставляет средства для учета и управления техническим оборудованием, инструментами и материалами. Это важно для эффективного планирования обслуживания и запасов.
* Техническая поддержка 24/7: Преимущества: Maxpanda предлагает техническую поддержку в режиме 24/7, что обеспечивает надежную помощь пользователям в случае возникновения проблем или необходимости консультации.
* Недостатки: Дополнительные модули: Для использования некоторых функций, пользователи могут потребовать дополнительные модули, что может повлиять на общую стоимость использования.

В целом, Maxpanda представляет собой привлекательное решение для управления техническим оборудованием и ресурсами. Простой интерфейс и возможность управления материалами делают его эффективным инструментом для автосервисов, однако стоит обратить внимание на возможные дополнительные расходы на модули.

Анализ существующих аналогов выявил разнообразие решений с разными функциональными возможностями. Многие CMS и CRM системы обладают функционалом, схожим с потребностями автосервисов, но часто они предназначены для более общего использования. Проектируемая база данных для автосервиса будет уникальным решением, сфокусированным на интегрированной поддержке всех аспектов работы в данной предметной области.

1. ФОРМИРОВАНИЕ ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ
   1. Общие требования

Общие функциональные требования для frontend и backend [1]:

* интуитивно понятный, красивый интерфейс для клиентской части;
* регистрация клиента;
* заказ на работу со стороны клиента;
* редактирование аккаунта пользователя (электронная почта, имя, фамилия, пароль);
* верификация почты, если клиент неактивный;
* полный просмотр информации о собственных заказах, и связанных с ними информацией;
* сброс пароля по электронной почте;
* удаление аккаунта пользователя;
* реквизиты оплаты работ через личный кабинет;
* связь с сервисом через приложение;
* рабочий имеет полный доступ к информации, связанной с ним (смены, штрафы, переводы заработной платы и т.д.);
* просмотр информации о не распределенных работах, возможность взять сверхурочные;
* оператор имеет доступ ко всем данным клиентов, для более тесной работы с нимим;
* оператор может давать обратную связь клиентам;
* менеджер назначает рабочих на смены;
* менеджер может регистрировать рабочих и операторов;
* менеджер производит закупку оборудования и запчастей;
* администратор имеет доступ ко всем данным, связанными с финансами;
* администратор оплачивает налоги и прочие расходы;
* администратор имеет полный доступ ко всему функцианалу и всем данным;
* администратор распоряжается добавлением и удалением скидок на работы;
* директор имеет полный доступ ко всему, как и администратор.
  1. Диаграмма вариантов использования

Функциональную модель предметной области представим в виде диаграммы вариантов использования, представляющей систему в виде набора вариантов использования, и пользователей, взаимодействующих с ним [2].

Диаграмма вариантов использования представлена на рисунке 2.1.

## 

Рисунок 2.1 – Диаграмма вариантов использования

* 1. Требования к качесту приложения

Требования к качеству приложения включают следующие пункты:

* Приложение должно обеспечивать загрузку данных при скорости интернета 12 Мбит/сек не более чем за секунду, что может достигатся путем разбиения данных на страницы.
* Приложение должно обрабатывать вводимые данные на русском и английском языках.
* Приложение должно корректно вести себя при неправильно введенных данных.
* Приложение должно завершать свою работу только корректным образом.
* Приложение должно корректно отрабатывать при возникновении ошибок и исключительных ситуаций.

1. ВЫБОР ИНСТРУМЕНТОВ РАЗРАБОТКИ
   1. Реляционные базы данных

Реляционная база данных – это набор данных с предопределенными связями между ними. Эти данные организованы в виде набора таблиц, состоящих из столбцов и строк. В таблицах хранится информация об объектах, представленных в базе данных. В каждом столбце таблицы хранится определенный тип данных, в каждой ячейке – значение атрибута. Каждая строка таблицы представляет собой набор связанных значений, относящихся к одному объекту или сущности. Каждая строка в таблице может быть помечена уникальным идентификатором, называемым первичным ключом, а строки из нескольких таблиц могут быть связаны с помощью внешних ключей. К этим данным можно получить доступ многими способами, и при этом реорганизовывать таблицы БД не требуется.

SQL (Structured Query Language) – основной интерфейс работы с реляционными базами данных. SQL стал стандартом Национального института стандартов США (ANSI) в 1986 году. Стандарт ANSI SQL поддерживается всеми популярными ядрами реляционных БД. Некоторые из ядер также включают расширения стандарта ANSI SQL, поддерживающие специфичный для этих ядер функционал. SQL используется для добавления, обновления и удаления строк данных, извлечения наборов данных для обработки транзакций и аналитических приложений, а также для управления всеми аспектами работы базы данных.

Целостность данных – это полнота, точность и единообразие данных. Для поддержания целостности данных в реляционных БД используется ряд инструментов. В их число входят первичные ключи, внешние ключи, ограничения «Not NULL», «Unique», «Default» и «Check». Эти ограничения целостности позволяют применять практические правила к данным в таблицах и гарантировать точность и надежность данных. Большинство ядер БД также поддерживает интеграцию пользовательского кода, который выполняется в ответ на определенные операции в БД.

Транзакция в базе данных – это один или несколько операторов SQL, выполненных в виде последовательности операций, представляющих собой единую логическую задачу. Транзакция представляет собой неделимое действие, то есть она должна быть выполнена как единое целое и либо должна быть записана в базу данных целиком, либо не должен быть записан ни один из ее компонентов. В терминологии реляционных баз данных транзакция завершается либо действием COMMIT, либо ROLLBACK. Каждая транзакция рассматривается как внутренне связный, надежный и независимый от других транзакций элемент. Для соблюдения целостности данных все транзакции в БД должны соответствовать требованиям ACID, то есть быть атомарными, единообразными, изолированными и надежными.

Атомарность – это условие, при котором либо транзакция успешно выполняется целиком, либо, если какая-либо из ее частей не выполняется, вся транзакция отменяется. Единообразие – это условие, при котором данные, записываемые в базу данных в рамках транзакции, должны соответствовать всем правилам и ограничениям, включая ограничения целостности, каскады и триггеры. Изолированность необходима для контроля над согласованностью и гарантирует базовую независимость каждой транзакции. Надежность подразумевает, что все внесенные в базу данных изменения на момент успешного завершения транзакции считаются постоянными [3].

* 1. PostgreSQL

При проектировании базы данных для автосервиса было принято решение в пользу PostgreSQL, обусловленное несколькими ключевыми причинами, объединяющими достаточный функционал и высокое удобство использования.

PostgreSQL – свободная объектно-реляционная система управления базами данных (СУБД). PostgreSQL является одной из наиболее популярных систем управления базами данных. Сам проект postgresql эволюционировал из другого проекта, который назывался Ingres. Формально развитие postgresql началось еще в 1986 году. Тогда он назывался POSTGRES. А в 1996 году проект был переименован в PostgreSQL, что отражало больший акцент на SQL. И, собственно, 8 июля 1996 года состоялся первый релиз продукта [4].

Плюсы PostgreSQL:

* большое количество встроенных типов данных (в том числе JSON);
* пользовательские объекты;
* поддержка сторонних плагинов;
* разнообразие индексов B-дерево, хеш, GiST, GIN, BRIN, Bloom;
* современный планировщик запросов;
* многоверсионность. Multiversion Concurrency Control (MVCC);
* удобное приложение pgAdmin, позволяющее с легкостью проектировать архитектуру базы данных;
* партицирование;
* лицензия;
* ведение журнала с упреждающей записью;
* открытый исходный код;
* хорошая документация.

В итоге, PostgreSQL был выбран в качестве СУБД для автосервисной базы данных, объединяя в себе высокую функциональность, открытый исходный код и простоту использования, что соответствует требованиям эффективной и надежной системы управления данными в автосервисе.

1. ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

В данной главе представлен процесс проектирования базы данных для автосервиса, начиная с инфологической модели и заканчивая определением требований к входным данным и форматам их хранения. Проектирование базы данных является критическим этапом разработки, определяющим структуру данных, их взаимосвязи и способы обработки [5].

* 1. Инфологическая модель

Инфологическая модель представляет собой высокоуровневое описание данных, отражающее их структуру и взаимосвязи. В контексте автосервиса определены основные сущности и их атрибуты, а также связи между ними. Разработка инфологической модели позволяет понять основные элементы системы и их взаимодействие:

1. Body Styles (Тип кузова):

* name – название типа кузова.

1. Brands (Бренд):

* name – название бренда.

1. Models (Модель):

* name – название модели.
* brand – ссылка на таблицу Brands.
* body\_style – ссылка на таблицу Body Styles.

1. Roles (Роль):

* name – название роли.

1. Persons (Человек):

* full\_name – полное имя человека.
* phone – номер телефона.
* email – адрес электронной почты.
* login – логин от аккаунта.
* password – пароль от аккаунта.
* role – ссылка на табоицу Roles.

1. Providers (Поставщик):

* name – имя или название поставщика.
* phone – номер телефона.
* email – адрес электронной почты.

1. Clients (Клиент):

* last\_visit – дата последнего посещения.

1. Qualifications (Квалификация):

* description – описание квалификации.

1. Workers (Рабочий):

* salary – почасовая заработная плата.
* hire\_date – дата найма сотрудника.
* unpaid\_hours – не оплаченные рабочие часы.
* paid\_hours – оплаченные рабочие часы.
* persanal\_qualities – личные качества сотрудника.

1. Shifts (Сиена):

* starts\_at – время и дата начала смены.
* end\_at – время и дата окончания смены.
* required\_qualification – ссылка на таблицу Qualifications.
* worker – ссылка на таблицу Workers.

1. Equipment (Оборудование):

* name – название типа кузова.
* description – описание оборудования.
* last\_used – дата последнего использования.
* expiery\_date – плановая дата замены.
* price – цена закупки единицы оборудования.
* required\_qualification – ссылка на таблицу Qualifications.

1. Services (Услуга):

* name – наименование услуги.
* description – описание услуги.
* price – цена.
* workload – трудоемкость работ.

1. Cars (Машина):

* client – ссылка на таблицу Clients.
* model – ссылка на таблицу Models.

1. Cash Accounts (Банковский счет):

* number – номер счета.
* person – ссылка на таблицу Persons.

1. Cash Transfers (Банковский перевод):

* amount – сумма перевода.
* account – ссылка на таблицу Cash Accounts.
* date – дата перевода.
* type – тип перевода.

1. Taxes (Уплаченный налог):

* name – название типа кузова.
* amount – сумма налога.
* transfer – ссылка на таблицу Cash Transfers.

1. Discounts (Скидка):

* service – ссылка на таблицу Services.
* fctor – множитель.

1. Reviews (Отзыв):

* review – отзыв.
* person – ссылка на таблицу Persons.

1. Penalties (Штраф):

* worker – ссылка на таблицу Workers.
* description – описание причины штрафа.
* date – дата штрафования.
* amount – размер штрафа.

1. Expenses (Расходы):

* description – описание расхода.
* transfer – ссылка на таблицу Cash Transfers.

1. Documents (Договор):

* name – название типа кузова.
* contract – текст договора.
* start\_at – дата начала работ.
* end\_at – дата окончания работ.
* car – ссылка на таблицу Cars.

1. Surcharges (Допалата):

* description – причина доплаты.
* document – ссылка на таблицу Documents.
* amount – сумма доплаты.

1. Breakdowns (Поломки):

* price – цена устранения.
* workload – трудоемкость.
* document – ссылка на таблицу Documents.
* description – описание поломки.

1. Defects (Дефект):

* descryption – описание дефекта.
* document – ссылка на таблицу Documents.

1. Transportation\_Costs (Транспортные расходы):

* document – ссылка на таблицу Documents.
* description – описание расходов.
* price – цена расхода.

1. Components (Деталь машины):

* name – название детали.
* price – цена закупки детали.
* date\_of\_manufacture – дата изготовления детали.
* proviser – ссылка на таблицу Providers.
* document – ссылка на таблицу Documents.
* serial\_number – серийный номер детали.
  1. Даталогическая Модель

Даталогическая модель представляет собой более конкретное описание базы данных, включая определение таблиц, полей, индексов и ключей. Каждая сущность из инфологической модели трансформирована в соответствующую таблицу, а атрибуты – в поля. Рассмотрены типы данных, ограничения целостности и связи между таблицами (см. рис. 4.1, рис 4.2).

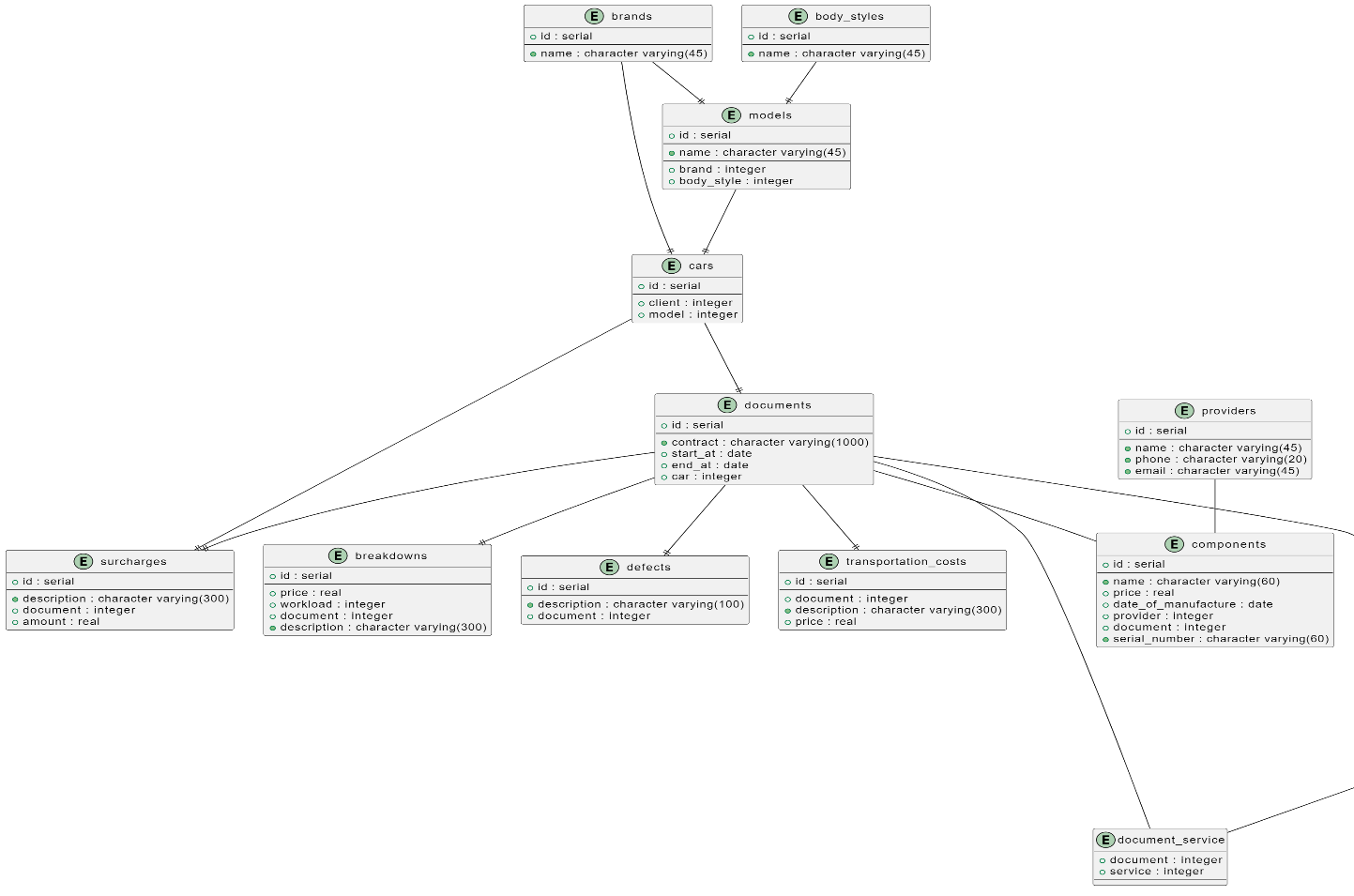


Рисунок 4.1 – Левая часть датологической модели базы данных

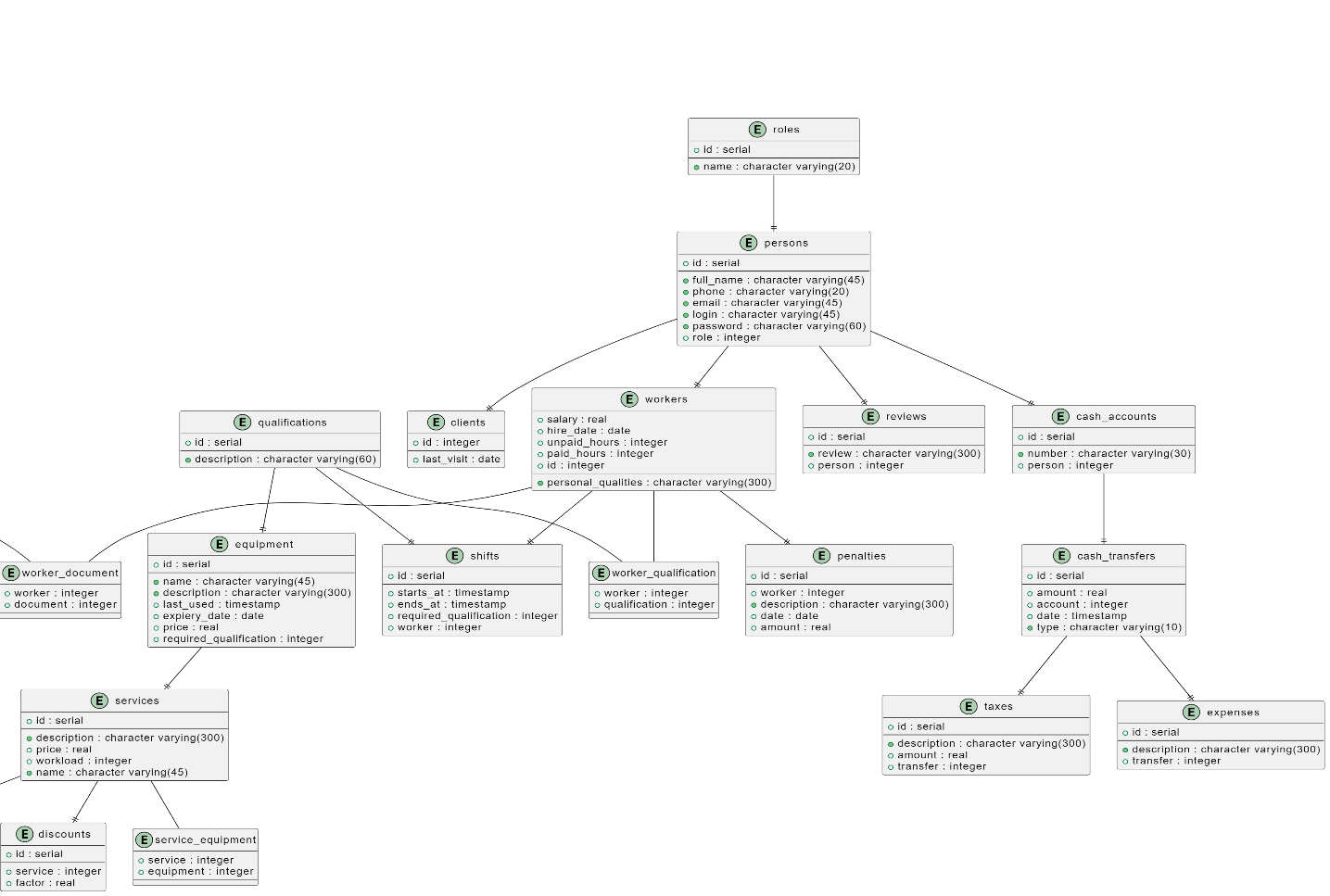


Рисунок 4.2 – Правая часть датологической модели базы данных

* 1. Диаграммы потока данных

В современных информационных системах и процессах проектирования баз данных ключевую роль играет визуализация потоков данных.

Эти диаграммы служат ключевым инструментом для визуализации общей структуры и взаимодействия сущностей в информационной системе.

Концептуальная диаграмма потоков данных представлена на рисунках 4.3, 4.4.

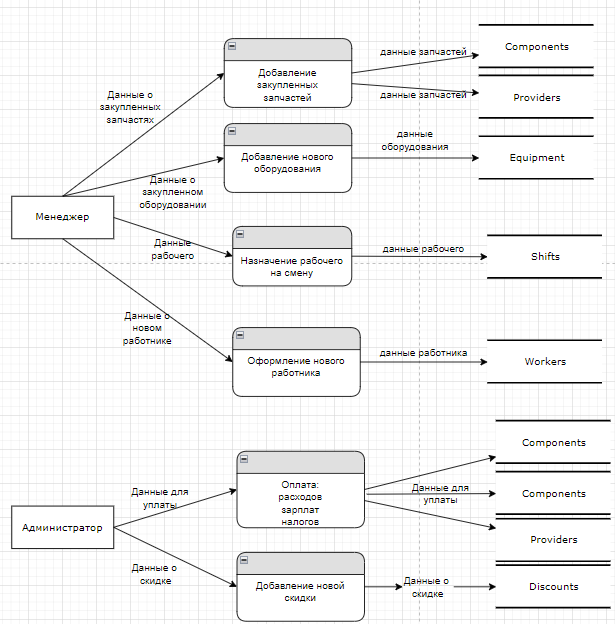


Рисунок 4.3 – Диаграмма потоков данных для менеджера и администратора

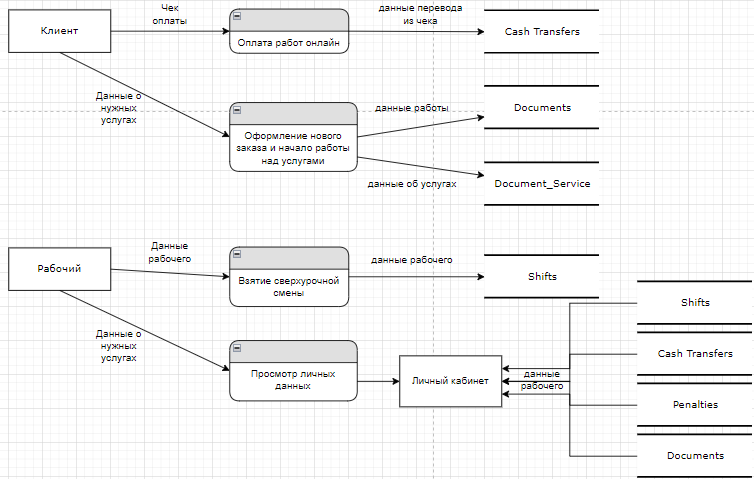


Рисунок 4.4 – Диаграмма потоков данных для клиента и рабочего

1. РАЗРАБОТКА БАЗЫ ДАННЫХ
   1. Создание сущностей базы данных

После того, как были написаны инфологические и даталогические модели базы данных, выбрана база данных и язык, можно приступать к непосредственной реализации сущностей базы данных.

Программный SQL-код, реализующий данные сущности представлен в приложении к данной пояснительной записке. Он реализует сущности согласно инфологической и даталогической моделям данных.

* 1. Создание хранимых процедур и функций

Все хранимые процедуры и функции, описанные в данном разделе также представлены в приложении А к пояснительной записке.

5.2.1 Процедура add\_model

Данная процедура предназначена для добавления модели машины. Процедура добавляет набор из трех связанных элементов: модель, тип кузова и бренд.

5.2.2 Процедура add\_job

Процедура предназначена для добавления нового заказа на работк. Входящими данными в данную процедуру являются все данные клиента, его машины и работы.

5.2.3 Процедура add\_worker

Процедура предназначена для добавления нового работника. Входящими данными в данную процедуру являются все данные работника.

5.2.4 Процедура pay\_salary

Процедура предназначена для выплаты заработной платы сотруднику. Создает перевод, а также обновляет неоплаченные и оплаченные часы. Входными данными является идентификатор рабочего.

5.2.5 Функция calculate\_total\_cost

В данной функции происходит подсчет общей суммы работ. Входными данными является идентификатор заказа на работу.

5.2.6 Функция get\_document\_info

Функция возвращает сгруппированную инормацию о конкретной работе. Входными данными является идентификатор заказа на работу.

5.2.7 Функция get\_document\_costs

Функция возвращает список сумм и описания для клиента, чтобы понимать, за что он платит.

5.2.8 Функция add\_component\_to\_document

Функция для занесения данных о купленных компонентах, Входними данными являются идентификатор заказа на работу и данные запчасти.

* 1. Создание триггеров

Программная реализация триггеров в виде sql-кода представлена в приложении А.

5.3.1 Триггер shift\_update\_unpaid\_hours

Триггер срабатывает после вставки в таблицу смен. Он добавляет неоплаченные часы в размере смены.

5.3.2 Триггер no\_delete\_role

Триггер срабатывает после удаления из таблицы ролей. Он не дает удалить роль.

* 1. Тестирование работоспособности базы данных

Для тестирования работоспособности базы данных был написан sql-скрипт для заполнения базы данных тестовыми данными, а так же sql-скрипт с вызовами функций и вставками в таблицы, полностью охватывающий все таблицы базы данных С sql-скрипатми можно ознакомиться в приложении B.

ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы была спроектирована и создана модель данных в рамках предметной области для приложения «Станция технического обслуживания».

В ходе выполнения работы было произведено сравнение основных аспектов реляционных и не реляционных баз данных, основных их свойств и особенностей. Была выбрана база данных для реализации данного проекта – PostgreSQL, которая уже не первый год является стандартом разработки различных веб-приложений в компаниях по всему миру.

Помимо всего, был разработан список запросов к базе данных, реализованы триггеры, процедуры, функции, индексы, ограничения и другие вещи, поддерживающие работу базы данных в исправном состоянии, а также реализующие основной функционал.

Таким образом, цели данной курсовой работу могут считаться достигнутыми, а задачи – выполненными.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Коннолли, Т., & Бегг, К. (2014). Базы данных: практический подход к проектированию, реализации и управлению (6-е изд.). Pearson.
2. Чен, П. П. (1976). Модель "Сущность-Связь" - К единому представлению данных. ACM Transactions on Database Systems (TODS), 1(1), 9-36. [Электронный ресурс]. – Режим доступа: ссылка.
3. Амблер, С. У. (2004). Введение в диаграммы активности UML 2.0. [Электронный ресурс]. – Режим доступа: https://www.agilemodeling.com/artifacts/activityDiagram.htm
4. Хей, Д. К., & Хей, Д. С. (1996). Паттерны модели данных: Карта метаданных. Morgan Kaufmann.
5. Кодд, Э. Ф. (1970). Реляционная модель данных для крупных общих баз данных. Communications of the ACM, 13(6), 377-387. [Электронный ресурс]. – Режим доступа: https://mipt.ru/dnbic/content/db.pdf.

**ПРИЛОЖЕНИЕ А**

**(информационное)**

# DDL скрипты

CREATE TABLE IF NOT EXISTS body\_styles

(

id serial PRIMARY KEY,

name character varying(45) NOT NULL

);

CREATE TABLE IF NOT EXISTS roles

(

id serial PRIMARY KEY,

name character varying(20) NOT NULL

);

CREATE TABLE IF NOT EXISTS brands

(

id serial PRIMARY KEY,

name character varying(45) NOT NULL

);

CREATE TABLE IF NOT EXISTS models

(

id serial PRIMARY KEY,

name character varying(45) NOT NULL,

brand integer NOT NULL,

body\_style integer NOT NULL,

FOREIGN KEY (brand)

REFERENCES brands (id)

ON DELETE CASCADE,

FOREIGN KEY (body\_style)

REFERENCES body\_styles (id)

ON DELETE CASCADE

);

CREATE TABLE IF NOT EXISTS persons

(

id serial PRIMARY KEY,

full\_name character varying(45) NOT NULL,

phone character varying(20) NOT NULL,

email character varying(45) NOT NULL,

login character varying(45) NOT NULL,

password character varying(60) NOT NULL,

role integer NOT NULL,

FOREIGN KEY (role)

REFERENCES roles (id)

ON DELETE CASCADE

);

CREATE TABLE IF NOT EXISTS providers

(

id serial PRIMARY KEY,

name character varying(45) NOT NULL,

phone character varying(20),

email character varying(45)

);

CREATE TABLE IF NOT EXISTS clients

(

id integer PRIMARY KEY NOT NULL,

last\_visit date NOT NULL,

FOREIGN KEY (id)

REFERENCES persons (id)

ON DELETE CASCADE

);

CREATE TABLE IF NOT EXISTS qualifications

(

id serial PRIMARY KEY NOT NULL,

description character varying(60)

);

CREATE TABLE IF NOT EXISTS workers

(

salary real NOT NULL,

hire\_date date NOT NULL,

unpaid\_hours integer NOT NULL,

paid\_hours integer NOT NULL,

personal\_qualities character varying(300),

id integer PRIMARY KEY NOT NULL,

FOREIGN KEY (id)

REFERENCES persons (id)

ON DELETE CASCADE

);

CREATE TABLE IF NOT EXISTS shifts

(

id serial PRIMARY KEY NOT NULL,

starts\_at timestamp NOT NULL,

ends\_at timestamp NOT NULL,

required\_qualification integer NOT NULL,

worker integer,

FOREIGN KEY (required\_qualification)

REFERENCES qualifications (id)

ON DELETE CASCADE,

FOREIGN KEY (worker)

REFERENCES workers (id)

ON DELETE CASCADE

);

CREATE TABLE IF NOT EXISTS equipment

(

id serial PRIMARY KEY,

name character varying(45) NOT NULL,

description character varying(300) NOT NULL,

last\_used timestamp NOT NULL,

expiery\_date date NOT NULL,

price real NOT NULL,

required\_qualification integer NOT NULL,

FOREIGN KEY (required\_qualification)

REFERENCES qualifications (id)

ON DELETE CASCADE

);

CREATE TABLE IF NOT EXISTS services

(

id serial PRIMARY KEY,

description character varying(300) NOT NULL,

price real NOT NULL,

workload integer NOT NULL,

name character varying(45) NOT NULL

);

CREATE TABLE IF NOT EXISTS cars

(

id serial PRIMARY KEY,

client integer NOT NULL,

model integer NOT NULL,

FOREIGN KEY (client)

REFERENCES clients (id)

ON DELETE CASCADE,

FOREIGN KEY (model)

REFERENCES models (id)

ON DELETE CASCADE

);

CREATE TABLE IF NOT EXISTS cash\_accounts

(

id serial PRIMARY KEY,

number character varying(30) NOT NULL,

person integer,

FOREIGN KEY (person)

REFERENCES persons (id)

ON DELETE CASCADE

);

CREATE TABLE IF NOT EXISTS cash\_transfers

(

id serial PRIMARY KEY,

amount real NOT NULL,

account integer NOT NULL,

date timestamp NOT NULL,

type character varying(10) NOT NULL,

FOREIGN KEY (account)

REFERENCES cash\_accounts (id)

ON DELETE CASCADE

);

CREATE TABLE IF NOT EXISTS taxes

(

id serial PRIMARY KEY,

description character varying(300) NOT NULL,

amount real NOT NULL,

transfer integer NOT NULL,

FOREIGN KEY (transfer)

REFERENCES cash\_transfers (id)

ON DELETE CASCADE

);

CREATE TABLE IF NOT EXISTS discounts

(

id serial PRIMARY KEY,

service integer NOT NULL,

factor real NOT NULL,

FOREIGN KEY (service)

REFERENCES services (id)

ON DELETE CASCADE

);

CREATE TABLE IF NOT EXISTS reviews

(

id serial PRIMARY KEY,

review character varying(300) NOT NULL,

person integer,

FOREIGN KEY (person)

REFERENCES persons (id)

ON DELETE CASCADE

);

-- Штрафы работникам

CREATE TABLE IF NOT EXISTS penalties

(

id serial PRIMARY KEY,

worker integer NOT NULL,

description character varying(300) NOT NULL,

date date NOT NULL,

amount real NOT NULL,

FOREIGN KEY (worker)

REFERENCES workers (id)

ON DELETE CASCADE

);

-- Расходы на аренду, комуналку и т.д.

CREATE TABLE IF NOT EXISTS expenses

(

id serial PRIMARY KEY,

description character varying(300) NOT NULL,

transfer integer NOT NULL,

FOREIGN KEY (transfer)

REFERENCES cash\_transfers (id)

ON DELETE CASCADE

);

CREATE TABLE IF NOT EXISTS documents

(

id serial PRIMARY KEY,

contract character varying(1000),

start\_at date,

end\_at date,

car integer NOT NULL,

FOREIGN KEY (car)

REFERENCES cars (id)

ON DELETE CASCADE

);

-- Доплаты клиентами

CREATE TABLE IF NOT EXISTS surcharges

(

id serial PRIMARY KEY,

description character varying(300) NOT NULL,

document integer NOT NULL,

amount real NOT NULL,

FOREIGN KEY (document)

REFERENCES documents (id)

ON DELETE CASCADE

);

-- В услугах будет устранение найденных дефектов

CREATE TABLE IF NOT EXISTS breakdowns

(

id serial PRIMARY KEY,

price real NOT NULL,

workload integer NOT NULL,

document integer NOT NULL,

description character varying(300) NOT NULL,

FOREIGN KEY (document)

REFERENCES documents (id)

ON DELETE CASCADE

);

-- внешние дефекты и неисправности, не требующие починки, но за которые потом чел предъявит

CREATE TABLE IF NOT EXISTS defects

(

id serial PRIMARY KEY,

description character varying(100) NOT NULL,

document integer NOT NULL,

FOREIGN KEY (document)

REFERENCES documents (id)

ON DELETE CASCADE

);

CREATE TABLE IF NOT EXISTS transportation\_costs

(

id serial PRIMARY KEY,

document integer NOT NULL,

description character varying(300) NOT NULL,

price real NOT NULL,

FOREIGN KEY (document)

REFERENCES documents (id)

ON DELETE NO ACTION

);

CREATE TABLE IF NOT EXISTS document\_service

(

document integer NOT NULL,

service integer NOT NULL,

FOREIGN KEY (document)

REFERENCES documents (id)

ON DELETE NO ACTION,

FOREIGN KEY (service)

REFERENCES services (id)

ON DELETE NO ACTION

);

CREATE TABLE IF NOT EXISTS service\_equipment

(

service integer NOT NULL,

equipment integer NOT NULL,

FOREIGN KEY (service)

REFERENCES services (id)

ON DELETE NO ACTION,

FOREIGN KEY (equipment)

REFERENCES equipment (id)

ON DELETE NO ACTION

);

CREATE TABLE IF NOT EXISTS worker\_document

(

worker integer NOT NULL,

document integer NOT NULL,

FOREIGN KEY (worker)

REFERENCES workers (id)

ON DELETE NO ACTION,

FOREIGN KEY (document)

REFERENCES documents (id)

ON DELETE NO ACTION

);

CREATE TABLE IF NOT EXISTS worker\_qualification

(

worker integer NOT NULL,

qualification integer NOT NULL,

FOREIGN KEY (worker)

REFERENCES workers (id)

ON DELETE NO ACTION,

FOREIGN KEY (qualification)

REFERENCES qualifications (id)

ON DELETE NO ACTION

);

CREATE TABLE IF NOT EXISTS components

(

id serial PRIMARY KEY,

name character varying(60) NOT NULL,

price real NOT NULL,

date\_of\_manufacture date NOT NULL,

provider integer,

document integer NOT NULL,

serial\_number character varying(60) NOT NULL,

FOREIGN KEY (provider)

REFERENCES providers (id)

ON DELETE CASCADE,

FOREIGN KEY (document)

REFERENCES documents (id)

ON DELETE CASCADE

);

CREATE OR REPLACE PROCEDURE add\_model(

IN p\_brand\_name character varying(45),

IN p\_body\_style\_name character varying(45),

IN p\_model\_name character varying(45)

)

LANGUAGE plpgsql

AS $$

DECLARE

v\_brand\_id integer;

v\_body\_style\_id integer;

v\_model\_id integer;

BEGIN

-- Проверка и добавление бренда

SELECT id INTO v\_brand\_id FROM brands WHERE name = p\_brand\_name;

IF v\_brand\_id IS NULL THEN

INSERT INTO brands (name) VALUES (p\_brand\_name) RETURNING id INTO v\_brand\_id;

END IF;

-- Проверка и добавление типа кузова

SELECT id INTO v\_body\_style\_id FROM body\_styles WHERE name = p\_body\_style\_name;

IF v\_body\_style\_id IS NULL THEN

INSERT INTO body\_styles (name) VALUES (p\_body\_style\_name) RETURNING id INTO v\_body\_style\_id;

END IF;

-- Проверка и добавление модели

SELECT id INTO v\_model\_id FROM models WHERE name = p\_model\_name AND brand = v\_brand\_id AND body\_style = v\_body\_style\_id;

IF v\_model\_id IS NULL THEN

INSERT INTO models (name, brand, body\_style) VALUES (p\_model\_name, v\_brand\_id, v\_body\_style\_id) RETURNING id INTO v\_model\_id;

END IF;

END;

$$;

CREATE OR REPLACE PROCEDURE add\_job(

IN p\_client\_full\_name character varying(45),

IN p\_client\_phone character varying(20),

IN p\_client\_email character varying(45),

IN p\_login character varying(45),

IN p\_password character varying(60),

IN p\_model\_name character varying(45),

IN p\_brand\_name character varying(45),

IN p\_body\_style\_name character varying(45),

IN p\_services\_ids integer[],

IN p\_contract character varying(1000),

IN p\_start\_at date,

IN p\_end\_at date

)

LANGUAGE plpgsql

AS $$

DECLARE

v\_client\_id integer;

v\_car\_id integer;

v\_model\_id integer;

v\_document\_id integer;

v\_service\_id integer;

BEGIN

-- Проверка и добавление клиента в persons

SELECT id INTO v\_client\_id FROM persons WHERE phone = p\_client\_phone;

IF v\_client\_id IS NULL THEN

-- Создание нового клиента, если его еще нет

INSERT INTO persons (full\_name, phone, email, login, password, role)

VALUES (p\_client\_full\_name, p\_client\_phone, p\_client\_email, p\_login, p\_password, (SELECT id FROM roles WHERE name = 'Клиент'))

RETURNING id INTO v\_client\_id;

END IF;

-- Проверка и добавление клиента в clients

INSERT INTO clients (id, last\_visit) VALUES (v\_client\_id, CURRENT\_DATE) ON CONFLICT (id) DO NOTHING;

-- Проверка и добавление машины

SELECT cars.id INTO v\_car\_id FROM cars

JOIN models ON cars.model = models.id

JOIN brands ON models.brand = brands.id

JOIN body\_styles ON models.body\_style = body\_styles.id

WHERE brands.name = p\_brand\_name AND models.name = p\_model\_name AND body\_styles.name = p\_body\_style\_name AND cars.client = v\_client\_id;

IF v\_car\_id IS NULL THEN

-- Создание новой машины, если ее еще нет

call add\_model(p\_brand\_name, p\_body\_style\_name, p\_model\_name);

SELECT models.id INTO v\_model\_id FROM models

JOIN brands ON models.brand = brands.id

JOIN body\_styles ON models.body\_style = body\_styles.id

WHERE brands.name = p\_brand\_name AND models.name = p\_model\_name AND body\_styles.name = p\_body\_style\_name;

INSERT INTO cars (client, model) VALUES (v\_client\_id, v\_model\_id) RETURNING id INTO v\_car\_id;

END IF;

-- Добавление записи о работе в documents

INSERT INTO documents (contract, start\_at, end\_at, car) VALUES (p\_contract, p\_start\_at, p\_end\_at, v\_car\_id) RETURNING id INTO v\_document\_id;

-- Проверка и добавление услуг

FOREACH v\_service\_id IN ARRAY p\_services\_ids

LOOP

-- Связываем услугу с документом

INSERT INTO document\_service (document, service) VALUES (v\_document\_id, v\_service\_id);

END LOOP;

END;

$$;

CREATE OR REPLACE PROCEDURE add\_worker(

IN p\_full\_name character varying(45),

IN p\_phone character varying(20),

IN p\_email character varying(45),

IN p\_login character varying(45),

IN p\_password character varying(60),

IN p\_salary real,

IN p\_hire\_date date,

IN p\_personal\_qualities character varying(300),

IN p\_qualifications character varying[]

)

LANGUAGE plpgsql

AS $$

DECLARE

v\_worker\_id integer;

v\_qualification\_name character varying(60);

BEGIN

-- Проверка существования телефона в persons

SELECT id INTO v\_worker\_id FROM persons WHERE phone = p\_phone;

-- Если не существует, добавляем рабочего

IF v\_worker\_id IS NULL THEN

-- Добавление информации о рабочем

INSERT INTO persons (full\_name, phone, email, login, password, role)

VALUES (p\_full\_name, p\_phone, p\_email, p\_login, p\_password, (SELECT id FROM roles WHERE name = 'Рабочий'))

RETURNING id INTO v\_worker\_id;

-- Добавление информации о рабочем

INSERT INTO workers (id, salary, hire\_date, unpaid\_hours, paid\_hours, personal\_qualities)

VALUES (v\_worker\_id, p\_salary, p\_hire\_date, 0, 0, p\_personal\_qualities);

-- Добавление квалификаций

FOREACH v\_qualification\_name IN ARRAY p\_qualifications

LOOP

-- Проверка существования квалификации

IF NOT EXISTS (SELECT 1 FROM qualifications WHERE description = v\_qualification\_name) THEN

-- Добавление новой квалификации

INSERT INTO qualifications (description) VALUES (v\_qualification\_name);

END IF;

-- Связываем квалификацию с рабочим

INSERT INTO worker\_qualification (worker, qualification)

VALUES (v\_worker\_id, (SELECT id FROM qualifications WHERE description = v\_qualification\_name));

END LOOP;

END IF;

END;

$$;

CREATE OR REPLACE PROCEDURE pay\_salary(IN p\_worker\_id integer)

LANGUAGE plpgsql

AS $$

DECLARE

v\_unpaid\_hours integer;

v\_hourly\_rate real;

v\_amount real;

BEGIN

-- Get the unpaid hours and hourly rate for the worker

SELECT unpaid\_hours, salary INTO v\_unpaid\_hours, v\_hourly\_rate

FROM workers

WHERE id = p\_worker\_id;

-- Calculate the amount to be paid

v\_amount := v\_unpaid\_hours \* v\_hourly\_rate;

-- Insert a cash transfer record for the payment

INSERT INTO cash\_transfers (amount, account, date, type)

VALUES (v\_amount, (SELECT id FROM cash\_accounts WHERE person = p\_worker\_id), NOW(), 'Salary');

-- Update the worker's paid and unpaid hours

UPDATE workers

SET paid\_hours = paid\_hours + v\_unpaid\_hours, unpaid\_hours = 0

WHERE id = p\_worker\_id;

END;

$$;

CREATE OR REPLACE FUNCTION calculate\_total\_cost(p\_document\_id INTEGER)

RETURNS REAL AS $$

DECLARE

v\_total\_cost REAL := 0;

v\_tmp REAL := 0;

BEGIN

-- Calculate total cost from document\_service

SELECT COALESCE(SUM(services.price), 0)

INTO v\_tmp

FROM document\_service

JOIN services ON document\_service.service = services.id

WHERE document\_service.document = p\_document\_id;

v\_total\_cost := v\_total\_cost + v\_tmp;

-- Add cost from breakdowns

SELECT COALESCE(SUM(price), 0)

INTO v\_tmp

FROM breakdowns

WHERE document = p\_document\_id;

v\_total\_cost := v\_total\_cost + v\_tmp;

-- Add cost from surcharges

SELECT COALESCE(SUM(amount), 0)

INTO v\_tmp

FROM surcharges

WHERE document = p\_document\_id;

v\_total\_cost := v\_total\_cost + v\_tmp;

-- Add cost from transportation\_costs

SELECT COALESCE(SUM(price), 0)

INTO v\_tmp

FROM transportation\_costs

WHERE document = p\_document\_id;

v\_total\_cost := v\_total\_cost + v\_tmp;

-- Add cost from components

SELECT COALESCE(SUM(price), 0)

INTO v\_tmp

FROM components

WHERE document = p\_document\_id;

v\_total\_cost := v\_total\_cost + v\_tmp;

RETURN v\_total\_cost;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION get\_document\_info(p\_document\_id INTEGER)

RETURNS TABLE (

document\_id INTEGER,

contract VARCHAR(1000),

start\_at DATE,

end\_at DATE,

car\_id INTEGER,

client\_id INTEGER,

client\_name VARCHAR(45),

client\_phone VARCHAR(20),

client\_email VARCHAR(45),

model\_name VARCHAR(45),

brand\_name VARCHAR(45),

body\_style\_name VARCHAR(45),

total\_cost REAL

) AS $$

BEGIN

RETURN QUERY

SELECT

d.id AS document\_id,

d.contract,

d.start\_at,

d.end\_at,

d.car AS car\_id,

c.client,

p.full\_name AS client\_name,

p.phone AS client\_phone,

p.email AS client\_email,

m.name AS model\_name,

b.name AS brand\_name,

bs.name AS body\_style\_name,

calculate\_total\_cost(d.id) AS total\_cost

FROM documents d

JOIN cars c ON d.car = c.id

JOIN persons p ON c.client = p.id

JOIN models m ON c.model = m.id

JOIN brands b ON m.brand = b.id

JOIN body\_styles bs ON m.body\_style = bs.id

WHERE d.id = p\_document\_id;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION get\_document\_costs(p\_document\_id INTEGER)

RETURNS TABLE (

description VARCHAR(300),

amount REAL

) AS $$

BEGIN

RETURN QUERY

SELECT

s.name AS description,

s.price AS amount

FROM document\_service ds

JOIN services s ON ds.service = s.id

WHERE ds.document = p\_document\_id

UNION

SELECT

b.description AS description,

b.price AS amount

FROM breakdowns b

WHERE b.document = p\_document\_id

UNION

SELECT

su.description AS description,

su.amount AS amount

FROM surcharges su

WHERE su.document = p\_document\_id

UNION

SELECT

tc.description AS description,

tc.price AS amount

FROM transportation\_costs tc

WHERE tc.document = p\_document\_id

UNION

SELECT

co.name AS description,

co.price AS amount

FROM components co

WHERE co.document = p\_document\_id;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION add\_component\_to\_document(

p\_document\_id INTEGER,

p\_name VARCHAR(60),

p\_price REAL,

p\_date\_of\_manufacture DATE,

p\_provider\_name VARCHAR(45),

p\_serial\_number VARCHAR(60)

)

RETURNS INTEGER AS $$

DECLARE

v\_provider\_id INTEGER;

v\_component\_id INTEGER;

BEGIN

IF p\_provider\_name != '' THEN

-- Check if the provider exists, and insert if not

SELECT id INTO v\_provider\_id

FROM providers

WHERE name = p\_provider\_name;

IF v\_provider\_id IS NULL THEN

INSERT INTO providers (name) VALUES (p\_provider\_name) RETURNING id INTO v\_provider\_id;

END IF;

END IF;

-- Insert the component

INSERT INTO components (name, price, date\_of\_manufacture, provider, document, serial\_number)

VALUES (p\_name, p\_price, p\_date\_of\_manufacture, v\_provider\_id, p\_document\_id, p\_serial\_number)

RETURNING id INTO v\_component\_id;

RETURN v\_component\_id;

END;

$$ LANGUAGE plpgsql;

-- Trigger function to update unpaid\_hours on shift insert or update

CREATE OR REPLACE FUNCTION update\_unpaid\_hours()

RETURNS TRIGGER AS $$

BEGIN

IF TG\_OP = 'INSERT' THEN

-- Add shift hours to the new worker's unpaid\_hours if not null

IF NEW.worker IS NOT NULL THEN

UPDATE workers

SET unpaid\_hours = unpaid\_hours + EXTRACT(HOUR FROM NEW.ends\_at - NEW.starts\_at)

WHERE id = NEW.worker;

END IF;

ELSIF TG\_OP = 'UPDATE' THEN

-- Subtract hours from the old worker's unpaid\_hours if not null

IF OLD.worker IS NOT NULL THEN

UPDATE workers

SET unpaid\_hours = unpaid\_hours - EXTRACT(HOUR FROM OLD.ends\_at - OLD.starts\_at)

WHERE id = OLD.worker;

END IF;

-- Add shift hours to the new worker's unpaid\_hours if not null

IF NEW.worker IS NOT NULL THEN

UPDATE workers

SET unpaid\_hours = unpaid\_hours + EXTRACT(HOUR FROM NEW.ends\_at - NEW.starts\_at)

WHERE id = NEW.worker;

END IF;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

-- Trigger to call the trigger function on shift insert or update

CREATE TRIGGER shift\_update\_unpaid\_hours

AFTER INSERT OR UPDATE ON shifts

FOR EACH ROW

EXECUTE FUNCTION update\_unpaid\_hours();

CREATE FUNCTION no\_delete() RETURNS trigger AS $$

BEGIN

RETURN NULL;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER no\_delete\_role BEFORE DELETE OR UPDATE ON roles

FOR EACH ROW EXECUTE PROCEDURE no\_delete();

**ПРИЛОЖЕНИЕ B**

**(информационное)**

# Скрипты тестирования

-- Вставка данных в таблицу qualifications

INSERT INTO qualifications (description) VALUES

('Специалист по диагностике'),

('Слесарь-ремонтник'),

('Автомаляр'),

('Сварщик'),

('Мастер по кузовному ремонту'),

('Электрик'),

('Монтажник охранных систем'),

('Технолог по покраске'),

('Шумоизолятор'),

('Техник по тонировке стекол'),

('Электромеханик');

-- Вставка данных в таблицу equipment

INSERT INTO equipment (name, description, last\_used, expiery\_date, price, required\_qualification) VALUES

('Диагностический сканер', 'Современный сканер для диагностики автомобиля', '2023-12-01 10:30:00', '2025-01-01', 5000.00, 1),

('Набор инструментов для ремонта', 'Профессиональный инструмент для проведения слесарных работ', '2023-11-15 14:45:00', '2024-12-31', 3000.00, 2),

('Покрасочная камера', 'Специализированная камера для покраски автомобилей', '2023-10-20 16:00:00', '2024-11-30', 10000.00, 8),

('Сварочный аппарат', 'Электродный сварочный аппарат для проведения сварочных работ', '2023-11-05 12:00:00', '2024-10-31', 8000.00, 4),

('Монтажный стол', 'Стол для проведения монтажных работ на кузове', '2023-12-02 09:30:00', '2025-02-28', 6000.00, 5),

('Электрический тестер', 'Тестер для проверки электрических систем автомобиля', '2023-12-03 08:00:00', '2024-09-30', 2000.00, 6),

('Оборудование для установки охранных систем', 'Набор инструментов для установки сигнализаций', '2023-11-25 11:15:00', '2024-08-31', 3000.00, 7),

('Шумоизоляционный материал', 'Материал для проведения работ по шумоизоляции автомобиля', '2023-10-15 13:45:00', '2024-09-30', 1000.00, 9),

('Оборудование для тонировки стекол', 'Специализированное оборудование для тонировки стекол', '2023-12-02 10:30:00', '2024-11-30', 4000.00, 10),

('Антизапотеватель', 'Устройство для предотвращения запотевания стекол', '2023-11-10 09:00:00', '2024-10-31', 800.00, 11),

('Электронный мультиметр', 'Измерительный прибор для работы с электрическими системами', '2023-10-05 15:30:00', '2024-12-31', 1500.00, 6),

('Диагностический осциллограф', 'Осциллограф для детальной диагностики электронных систем', '2023-11-20 11:30:00', '2024-11-30', 7000.00, 1),

('Рихтовочный молоток', 'Профессиональный молоток для рихтовки кузова', '2023-10-10 14:15:00', '2024-09-30', 500.00, 2),

('Пневматический пистолет для покраски', 'Инструмент для равномерного нанесения краски', '2023-10-25 16:30:00', '2024-12-31', 2000.00, 8),

('Плазменная резка', 'Оборудование для точной плазменной резки', '2023-12-01 09:00:00', '2025-01-31', 10000.00, 4),

('Плита для выправки деталей', 'Специализированная плита для выправки деформаций', '2023-11-15 13:00:00', '2024-10-31', 3000.00, 5),

('Индукционный обогреватель', 'Оборудование для быстрого обогрева и выправки деталей', '2023-12-05 10:45:00', '2025-02-28', 4000.00, 5),

('Оптический выравниватель', 'Прибор для точного выравнивания кузовных элементов', '2023-10-30 15:15:00', '2024-11-30', 6000.00, 5),

('Электронный сварочный аппарат', 'Аппарат для точной электронной сварки', '2023-11-10 12:30:00', '2024-10-31', 6000.00, 4),

('Автомобильный подъемник', 'Подъемник для поднятия автомобиля', '2023-10-18 14:00:00', '2024-09-30', 7000.00, 2),

('Лазерный регулировщик фар', 'Прибор для точной настройки автомобильных фар', '2023-11-05 11:30:00', '2024-12-31', 2500.00, 6),

('Инструмент для тонировки стекол', 'Специальный инструмент для проведения работ по тонировке стекол', '2023-12-02 15:30:00', '2024-11-30', 3000.00, 10);

INSERT INTO service\_equipment (service, equipment) VALUES

(1, 1), -- Диагностика и текущий слесарный ремонт -> Диагностический сканер

(1, 2), -- Диагностика и текущий слесарный ремонт -> Набор инструментов для ремонта

(2, 3), -- Капитальный слесарный ремонт -> Покрасочная камера

(2, 4), -- Капитальный слесарный ремонт -> Сварочный аппарат

(3, 5), -- Удаление вмятин и рихтовка -> Монтажный стол

(3, 6), -- Удаление вмятин и рихтовка -> Электрический тестер

(4, 7), -- Сварочные работы -> Оборудование для установки охранных систем

(5, 8), -- Покраска -> Шумоизоляционный материал

(6, 9), -- Установка охранных систем -> Сетка для защиты радиатора

(7, 10), -- Шумоизоляция авто -> Оборудование для тонировки стекол

(8, 11), -- Антидождь -> Антизапотеватель

(9, 12), -- Тонировка стекол -> Электронный мультиметр

(10, 13), -- Антизапотеватель -> Диагностический осциллограф

(11, 14), -- Продолжение вставки данных в таблицу equipment -> Рихтовочный молоток

(4, 15), -- Сварочные работы -> Плита для выправки деталей

(5, 16), -- Покраска -> Индукционный обогреватель

(6, 17), -- Установка охранных систем -> Оптический выравниватель

(7, 18), -- Шумоизоляция авто -> Электронный сварочный аппарат

(8, 19), -- Антидождь -> Автомобильный подъемник

(9, 20), -- Тонировка стекол -> Лазерный регулировщик фар

(10, 21), -- Антизапотеватель -> Инструмент для тонировки стекол

(11, 22); -- Продолжение вставки данных в таблицу equipment -> Антизапотеватель

-- Вставка данных в таблицу providers

INSERT INTO providers (name, phone, email) VALUES

('ООО "АвтоЗапчасти"', '+7 (123) 456-7890', 'info@avtozapchasti.ru'),

('ИП "Автоаксессуары"', '+7 (987) 654-3210', 'info@autoaksessuary.ru'),

('ТОО "АвтоСервис"', '+7 (111) 222-3333', 'info@autoservice.kz'),

('Группа компаний "Авто-Тех"', '+7 (555) 777-8888', 'info@auto-tech.com'),

('ООО "Автоэлектроника"', '+7 (999) 888-7777', 'info@autoelektronika.ru'),

('ООО "Запчасти-Люкс"', '+7 (777) 555-4444', 'info@zapchasti-lyux.ru'),

('ИП "АвтоРезерв"', '+7 (333) 222-1111', 'info@autoreserve.ru'),

('ТОО "АвтоМагазин"', '+7 (888) 999-0000', 'info@automagazin.kz'),

('Группа компаний "Авто-Мастер"', '+7 (111) 333-5555', 'info@auto-master.com'),

('ООО "АвтоПром"', '+7 (666) 555-4444', 'info@autoprom.ru');

-- Вставка данных в таблицу shifts

INSERT INTO shifts (starts\_at, ends\_at, required\_qualification, worker) VALUES

('2023-12-01 08:00:00', '2023-12-01 16:00:00', 2, 1),

('2023-12-02 12:00:00', '2023-12-02 20:00:00', 3, NULL),

('2023-12-03 09:00:00', '2023-12-03 17:00:00', 1, 2),

('2023-12-04 10:00:00', '2023-12-04 18:00:00', 4, NULL),

('2023-12-05 08:00:00', '2023-12-05 16:00:00', 2, 3),

('2023-12-06 14:00:00', '2023-12-06 22:00:00', 5, NULL),

('2023-12-07 11:00:00', '2023-12-07 19:00:00', 3, 4),

('2023-12-08 09:00:00', '2023-12-08 17:00:00', 1, NULL),

('2023-12-09 08:00:00', '2023-12-09 16:00:00', 2, 5),

('2023-12-10 12:00:00', '2023-12-10 20:00:00', 4, NULL),

('2023-12-11 08:00:00', '2023-12-11 16:00:00', 1, NULL),

('2023-12-12 14:00:00', '2023-12-12 22:00:00', 2, NULL),

('2023-12-13 09:00:00', '2023-12-13 17:00:00', 3, NULL),

('2023-12-14 11:00:00', '2023-12-14 19:00:00', 4, 1),

('2023-12-15 10:00:00', '2023-12-15 18:00:00', 5, NULL),

('2023-12-16 08:00:00', '2023-12-16 16:00:00', 1, 2),

('2023-12-17 12:00:00', '2023-12-17 20:00:00', 2, NULL),

('2023-12-18 09:00:00', '2023-12-18 17:00:00', 3, 3),

('2023-12-19 08:00:00', '2023-12-19 16:00:00', 4, NULL),

('2023-12-20 14:00:00', '2023-12-20 22:00:00', 5, NULL),

('2023-12-21 11:00:00', '2023-12-21 19:00:00', 1, 4),

('2023-12-22 10:00:00', '2023-12-22 18:00:00', 2, NULL),

('2023-12-23 08:00:00', '2023-12-23 16:00:00', 3, 5),

('2023-12-24 12:00:00', '2023-12-24 20:00:00', 4, NULL),

('2023-12-25 09:00:00', '2023-12-25 17:00:00', 5, 1),

('2023-12-26 08:00:00', '2023-12-26 16:00:00', 1, NULL),

('2023-12-27 14:00:00', '2023-12-27 22:00:00', 2, NULL),

('2023-12-28 09:00:00', '2023-12-28 17:00:00', 3, NULL),

('2023-12-29 11:00:00', '2023-12-29 19:00:00', 4, 2),

('2023-12-30 10:00:00', '2023-12-30 18:00:00', 5, NULL),

('2023-12-31 08:00:00', '2023-12-31 16:00:00', 1, 3),

('2024-01-01 12:00:00', '2024-01-01 20:00:00', 2, NULL),

('2024-01-02 09:00:00', '2024-01-02 17:00:00', 3, NULL),

('2024-01-03 08:00:00', '2024-01-03 16:00:00', 4, 4),

('2024-01-04 14:00:00', '2024-01-04 22:00:00', 5, NULL),

('2024-01-05 11:00:00', '2024-01-05 19:00:00', 1, NULL),

('2024-01-06 10:00:00', '2024-01-06 18:00:00', 2, 5),

('2024-01-07 08:00:00', '2024-01-07 16:00:00', 3, NULL),

('2024-01-08 12:00:00', '2024-01-08 20:00:00', 4, NULL),

('2024-01-09 09:00:00', '2024-01-09 17:00:00', 5, NULL);

-- Вставка данных в таблицу cash\_accounts

INSERT INTO cash\_accounts (number) VALUES

('123456789012345'),

('234567890123456'),

('345678901234567'),

('456789012345678'),

('456789012345673'),

('456789012345672'),

('456789012345671'),

('456789012345670'),

('456789012345679'),

('567890123456789');

-- Вставка данных в таблицу cash\_transfers

INSERT INTO cash\_transfers (amount, account, date, type) VALUES

(1000.00, 11, '2023-12-01 10:00:00', 'Taxes'),

(500.00, 12, '2023-12-02 12:30:00', 'Taxes'),

(2000.00, 13, '2023-12-03 15:45:00', 'Taxes'),

(1500.00, 14, '2023-12-04 09:20:00', 'Taxes'),

(800.00, 15, '2023-12-05 14:00:00', 'Taxes'),

(200.00, 11, '2023-12-06 09:30:00', 'Taxes'),

(350.00, 12, '2023-12-07 13:15:00', 'Taxes'),

(180.00, 13, '2023-12-08 16:45:00', 'Taxes'),

(450.00, 14, '2023-12-09 10:30:00', 'Taxes'),

(250.00, 15, '2023-12-10 12:00:00', 'Taxes'),

(200.00, 11, '2023-12-11 09:30:00', 'Taxes'),

(350.00, 12, '2023-12-12 13:15:00', 'Taxes'),

(180.00, 13, '2023-12-13 16:45:00', 'Taxes'),

(450.00, 14, '2023-12-14 10:30:00', 'Taxes'),

(250.00, 15, '2023-12-15 12:00:00', 'Taxes'),

(300.00, 11, '2023-12-16 14:30:00', 'Taxes'),

(180.00, 12, '2023-12-17 17:15:00', 'Taxes'),

(90.00, 13, '2023-12-18 09:45:00', 'Taxes'),

(100.00, 14, '2023-12-19 11:20:00', 'Taxes'),

(250.00, 15, '2023-12-20 14:00:00', 'Taxes'),

(300.00, 11, '2023-12-21 16:30:00', 'Taxes'),

(70.00, 12, '2023-12-22 19:15:00', 'Taxes'),

(130.00, 13, '2023-12-23 10:45:00', 'Taxes'),

(110.00, 14, '2023-12-24 12:20:00', 'Taxes'),

(90.00, 15, '2023-12-25 14:50:00', 'Taxes'),

(220.00, 11, '2023-12-26 17:30:00', 'Taxes'),

(200.00, 12, '2023-12-27 20:15:00', 'Taxes'),

(40.00, 13, '2023-12-28 09:00:00', 'Taxes'),

(120.00, 14, '2023-12-29 11:30:00', 'Taxes'),

(70.00, 15, '2023-12-30 14:00:00', 'Taxes'),

(60.00, 15, '2023-12-31 16:30:00', 'Taxes'),

(140.00, 11, '2024-01-01 19:15:00', 'Taxes'),

(80.00, 13, '2024-01-02 10:45:00', 'Taxes'),

(180.00, 12, '2024-01-03 12:20:00', 'Taxes'),

(150.00, 11, '2024-01-04 14:50:00', 'Taxes'),

(110.00, 11, '2024-01-05 17:30:00', 'Taxes'),

(120.00, 12, '2024-01-06 20:15:00', 'Taxes'),

(100.00, 13, '2024-01-07 09:00:00', 'Taxes'),

(50.00, 14, '2024-01-08 11:30:00', 'Taxes'),

(75.00, 15, '2024-01-09 14:00:00', 'Taxes');

-- Вставка данных в таблицу taxes

INSERT INTO taxes (description, amount, transfer) VALUES

('Налог на доходы', 150.00, 6),

('Налог на имущество', 75.00, 7),

('Налог на продажи', 300.00, 8),

('Налог на прибыль', 200.00, 9),

('Налог на транспорт', 100.00, 10),

('Земельный налог', 50.00, 11),

('Налог на добавленную стоимость (НДС)', 120.00, 12),

('Акцизный налог', 80.00, 13),

('Налог на прибыль организаций', 300.00, 14),

('Транспортный налог', 90.00, 15),

('Налог на недвижимость', 80.00, 16),

('Налог на прибыль физических лиц', 120.00, 17),

('Торговый сбор', 50.00, 18),

('Акциз на алкогольные напитки', 200.00, 19),

('Страховые взносы', 150.00, 20),

('Земельный налог', 60.00, 21),

('Налог на добавленную стоимость (НДС)', 180.00, 22),

('Транспортный налог', 90.00, 23),

('Налог на природопользование', 100.00, 24),

('Акциз на табачные изделия', 250.00, 25),

('Налог на игорный бизнес', 300.00, 26),

('Налог на использование водных объектов', 70.00, 27),

('Единый социальный налог', 130.00, 28),

('Налог на транспортные средства', 110.00, 29),

('Налог на профессиональный доход', 90.00, 30),

('Акциз на автотранспорт', 220.00, 31),

('Налог на добычу полезных ископаемых', 200.00, 32),

('Загрязнение окружающей среды', 40.00, 33),

('Налог на игровые автоматы', 120.00, 34),

('Страхование от несчастных случаев', 70.00, 35),

('Налог на хранение отходов', 60.00, 36),

('Единый налог на вмененный доход', 140.00, 37),

('Налог на вывоз отходов', 80.00, 38),

('Акциз на пиво', 180.00, 39),

('Налог на имущество предприятий', 150.00, 40),

('Транзитный налог', 110.00, 41),

('Налог на легковые автомобили', 120.00, 42),

('Единый налог на водные объекты', 100.00, 43),

('Налог на рекламу', 50.00, 44),

('Налог на использование объектов животного мира', 75.00, 45);

-- Вставка данных в таблицу discounts

INSERT INTO discounts (service, factor) VALUES

(1, 0.9),

(3, 0.8),

(5, 0.95),

(2, 0.85);

INSERT INTO reviews (review, person) VALUES

('Отличный сервис, всё сделали быстро и качественно!', 6),

('Немного задержались с ремонтом, но результат порадовал.', 7),

('Профессиональный подход, всё рассказали и показали.', 8),

('Цены не самые низкие, но качество работ заслуживает.', 9),

('Вежливый персонал, хороший сервис, рекомендую.', 10);

-- Вставка данных в таблицу penalties

INSERT INTO penalties (worker, description, date, amount) VALUES

(1, 'Опоздание на работу', '2023-12-01', 50.00),

(2, 'Невыполнение задания', '2023-12-02', 100.00),

(3, 'Использование ненадлежащего оборудования', '2023-12-03', 75.00),

(4, 'Нарушение техники безопасности', '2023-12-04', 120.00),

(5, 'Небрежное обращение с инструментами', '2023-12-05', 80.00);

INSERT INTO cash\_transfers (amount, account, date, type) VALUES

(1500.00, 16, '2023-12-10 10:00:00', 'Expense'),

(500.00, 17, '2023-12-11 12:30:00', 'Expense'),

(800.00, 18, '2023-12-12 15:45:00', 'Expense'),

(1200.00, 19, '2023-12-13 09:20:00', 'Expense'),

(1000.00, 20, '2023-12-14 14:00:00', 'Expense');

INSERT INTO expenses (description, transfer) VALUES

('Аренда офиса', 46),

('Коммунальные платежи', 47),

('Офисное оборудование', 48),

('Реклама и маркетинг', 49),

('Печеньки с чаем', 50);

-- Вставка данных в таблицу surcharges

INSERT INTO surcharges (description, document, amount) VALUES

('Доплата за ускоренное обслуживание', 1, 50.00),

('Доплата за использование оригинальных запчастей', 2, 100.00),

('Доплата за экспресс-доставку деталей', 3, 75.00),

('Доплата за дополнительные услуги', 4, 120.00),

('Доплата за гарантированный срок ремонта', 5, 80.00);

INSERT INTO breakdowns (price, workload, document, description) VALUES

(200.00, 10, 1, 'Замена поршневых колец и амортизаторов'),

(300.00, 15, 2, 'Замена тормозной системы и деталей'),

(150.00, 8, 3, 'Диагностика и устранение мелких неполадок'),

(400.00, 20, 4, 'Замена изношенных деталей'),

(250.00, 12, 5, 'Устранение вмятин и рихтовка кузова');

-- Вставка данных в таблицу defects

INSERT INTO defects (description, document) VALUES

('Мелкие царапины на кузове', 1),

('Незначительные потертости на лобовом стекле', 2),

('Легкие сколы краски', 3),

('Минимальные дефекты на бампере', 4),

('Мелкие царапины на двери', 5);

-- Вставка данных в таблицу transportation\_costs

INSERT INTO transportation\_costs (document, description, price) VALUES

(1, 'Транспортировка до сервиса', 50.00),

(2, 'Доставка запчастей', 30.00),

(3, 'Транспортировка автомобиля клиента', 40.00),

(4, 'Доставка деталей для ремонта', 35.00),

(5, 'Транспортировка до покрасочной мастерской', 60.00);

-- Вставка данных в таблицу components

INSERT INTO components (name, price, date\_of\_manufacture, provider, document, serial\_number) VALUES

('Двигатель', 3000.00, '2023-01-15', 1, 1, 'ENG123'),

('Тормозные колодки', 100.00, '2023-02-20', 2, 2, 'BRK456'),

('Фильтр масляный', 20.00, '2023-03-10', 3, 3, 'OIL789'),

('Аккумулятор', 150.00, '2023-04-05', 4, 4, 'BAT101'),

('Свечи зажигания', 15.00, '2023-05-12', 5, 5, 'SPK202');

call add\_model('Игнат', 'Игнат', 'Игнат');

CALL add\_job(

'John Doe', -- p\_client\_full\_name

'123-456-7890', -- p\_client\_phone

'john.doe@example.com', -- p\_client\_email

'client', -- login

'client', -- password

'Toyota Camry', -- p\_model\_name

'Toyota', -- p\_brand\_name

'Седан', -- p\_body\_style\_name

ARRAY[1, 2, 3], -- p\_services\_ids

'Contract for John Doe', -- p\_contract

'2023-01-01'::date, -- p\_start\_at

'2023-01-10'::date -- p\_end\_at

);

CALL add\_job(

'Alice Smith', -- p\_client\_full\_name

'987-654-3210', -- p\_client\_phone

'alice.smith@example.com', -- p\_client\_email

'client', -- login

'client', -- password

'Honda Accord', -- p\_model\_name

'Honda', -- p\_brand\_name

'Sedan', -- p\_body\_style\_name

ARRAY[4, 5], -- p\_services\_ids

'Contract for Alice Smith', -- p\_contract

'2023-02-15'::date, -- p\_start\_at

'2023-02-28'::date -- p\_end\_at

);

CALL add\_job(

'Bob Johnson', -- p\_client\_full\_name

'555-123-4567', -- p\_client\_phone

'bob.johnson@example.com', -- p\_client\_email

'client', -- login

'client', -- password

'Ford Mustang', -- p\_model\_name

'Ford', -- p\_brand\_name

'Coupe', -- p\_body\_style\_name

ARRAY[6, 7, 8], -- p\_services\_ids

'Contract for Bob Johnson', -- p\_contract

'2023-03-10'::date, -- p\_start\_at

'2023-03-20'::date -- p\_end\_at

);

CALL add\_worker(

'John Doe',

'+123456789',

'john.doe@example.com',

'johndoe',

'password123',

30.00,

'2023-01-15',

'Hardworking and dedicated',

ARRAY['Mechanic', 'Electrical Repairs']

);

CALL add\_worker(

'Alice Smith',

'+987654321',

'alice.smith@example.com',

'alicesmith',

'securepass',

35.00,

'2023-02-01',

'Experienced in diagnostics',

ARRAY['Diagnostic Specialist', 'Engine Tuning']

);

CALL add\_worker(

'Bob Johnson',

'+111222333',

'bob.johnson@example.com',

'bobjohnson',

'secretword',

28.00,

'2023-03-10',

'Attention to detail',

ARRAY['Oil Change', 'Tire Rotation']

);

call pay\_salary(1);

call pay\_salary(2);

call pay\_salary(3);

update shifts

set worker = 1

where id = 2;

SELECT calculate\_total\_cost(1);

SELECT \* FROM get\_document\_info(1);

SELECT \* FROM get\_document\_costs(1);

SELECT add\_component\_to\_document(1, 'CPU', 300.00, '2023-01-15', 'Intel', 'SN12345');

SELECT add\_component\_to\_document(2, 'RAM', 80.00, '2023-02-20', 'Corsair', 'SN67890');

SELECT add\_component\_to\_document(3, 'GPU', 500.00, '2023-03-25', 'NVIDIA', 'SNABCDE');