Частное учреждение профессионального образования
«Высшая школа предпринимательства»
(ЧУПО «ВШП»)

КУРСОВАЯ РАБОТА

«Разработка базы данных для системы управления прокатом электросамокатов»

Выполнил:
студент 3-го курса специальности
09.02.07 «Информационные системы
и программирование»
Заволокин Михаил Аркадьевич
подпись: _____

Проверил:
преподаватель дисциплины,
преподаватель ЧУПО «ВШП»,
к.ф.н. Ткачев П.С.
оценка: _____
подпись: _____

Тверь, 2025 г.

Содержание

Введение 3
Глава 1. Анализ бизнес-процессов и требований 5
1.1. Описание ключевых бизнес-процессов 5
1.2. Функциональные требования 6
1.4. Сравнение СУБД и обоснование выбора 8
Глава 2. Проектирование и реализация базы данных 11
2.1. Логическая модель данных 11
2.3. Наполнение тестовыми данными 17
2.4. Реализация бизнес-логики 18
2.5. Тестирование и оптимизация 18
Заключение 20
Приложение 23
Приложение 1. DDL-скрипты создания структуры базы данных 23
Приложение 2. Скрипты заполнения тестовыми данными 25
Приложение 3. Хранимые процедуры и функции 26
Приложение 4. Представления и триггеры 28
Приложение 5. Примеры SQL-запросов для основных операций 30

Введение

Современный мир характеризуется стремительным развитием технологий и изменением образа жизни людей. Одной из наиболее динамично развивающихся сфер является рынок микромобильности, включающий в себя различные виды персонального электротранспорта. Согласно данным исследования McKinsey & Company за 2024 год, мировой рынок микромобильности оценивается в 47,8 миллиарда долларов США и прогнозируется его рост до 96,7 миллиарда долларов к 2030 году.

Электросамокаты занимают особое место в экосистеме городского транспорта, предоставляя пользователям удобную, экологичную и экономически эффективную альтернативу традиционным способам передвижения. В России рынок каршеринга электросамокатов демонстрирует устойчивый рост: по данным аналитического агентства TelecomDaily, в 2024 году количество поездок на электросамокатах в крупных городах России увеличилось на 35% по сравнению с предыдущим годом.

Актуальность темы обусловлена необходимостью создания эффективных информационных систем для управления парком электросамокатов, обработки большого объема пользовательских данных и обеспечения бесперебойной работы сервиса. Качественно спроектированная база данных является основой для успешного функционирования таких систем.

Цель работы: разработать структуру базы данных для сервиса аренды электросамокатов, обеспечивающую эффективное хранение, обработку и управление данными.

Задачи работы:

1. Проанализировать бизнес-процессы сервиса аренды электросамокатов
2. Определить функциональные и нефункциональные требования к базе данных
3. Выбрать подходящую СУБД и обосновать выбор

4. Спроектировать логическую и физическую модели базы данных

1. Реализовать структуру базы данных с использованием DDL-команд

2. Создать тестовые данные и реализовать основную бизнес-логику

3. Протестировать производительность и предложить рекомендации по масштабированию

Объект исследования: информационные процессы в сервисе аренды электросамокатов.

Предмет исследования: структура и организация базы данных для хранения и обработки информации о пользователях, электросамокатах, арендах и платежах.

Методы исследования: системный анализ, моделирование данных, проектирование баз данных, тестирование производительности.

Глава 1. Анализ бизнес-процессов и требований

1.1. Описание ключевых бизнес-процессов

Сервис аренды электросамокатов представляет собой сложную систему, включающую множество взаимосвязанных процессов. Рассмотрим основные бизнес-процессы, которые должна поддерживать разрабатываемая база данных.

Процесс регистрации пользователей

Регистрация пользователя является отправной точкой взаимодействия с сервисом. Процесс включает сбор персональных данных пользователя: имени, фамилии, номера телефона, электронной почты, данных документа, удостоверяющего личность. Система должна обеспечивать уникальность пользователей по ключевым идентификаторам и валидацию введенных данных. Дополнительно создается пользовательский баланс для оплаты услуг аренды.

Процесс управления парком электросамокатов

Управление парком включает регистрацию новых электросамокатов в системе, отслеживание их текущего состояния, местоположения и технических характеристик. Каждый электросамокат имеет уникальный идентификатор, модель, уровень заряда батареи, GPS-координаты и статус доступности. Система должна поддерживать различные состояния самоката: доступен для аренды, в процессе аренды, на техническом обслуживании, неисправен.

Процесс поиска и бронирования электросамокатов

Пользователи должны иметь возможность находить доступные электросамокаты в определенном радиусе от своего местоположения. Система обрабатывает запросы на поиск, фильтрует доступные транспортные средства и предоставляет информацию о расстоянии до них, уровне заряда и стоимости аренды. Возможность предварительного бронирования самоката на ограниченное время также должна быть реализована.

Процесс начала и завершения аренды

Начало аренды включает проверку баланса пользователя, блокировку стоимости аренды, активацию самоката и начало отсчета времени. Система фиксирует точное время начала аренды, начальные GPS-координаты и состояние самоката. Завершение аренды предполагает расчет итоговой стоимости на основе времени использования и пройденного расстояния, списание средств с баланса пользователя и обновление статуса самоката.

Процесс обработки платежей

Система должна поддерживать различные способы пополнения баланса: банковские карты, электронные кошельки, бонусные программы. Необходимо вести историю всех финансовых операций, обеспечивать безопасность транзакций и предоставлять пользователям детальную информацию о расходах.

Процесс технического обслуживания

Регулярное техническое обслуживание электросамокатов включает зарядку батарей, проверку технического состояния, мелкий ремонт и перемещение самокатов в популярные локации. База данных должна отслеживать график обслуживания, фиксировать выполненные работы и их стоимость.

Процесс аналитики и отчетности

Система должна предоставлять возможность формирования различных отчетов: статистика использования самокатов, популярные маршруты, финансовые показатели, эффективность работы парка. Эти данные необходимы для принятия управленческих решений и оптимизации бизнес-процессов.

1.2. Функциональные требования

На основе анализа бизнес-процессов были определены следующие функциональные требования к базе данных:

Управление пользователями:

Регистрация новых пользователей с валидацией уникальности контактных данных

Хранение персональной информации и документов пользователей

Управление пользовательскими балансами и бонусными счетами

Ведение истории активности пользователей

Управление парком электросамокатов:

Регистрация электросамокатов с техническими характеристиками

Отслеживание текущего местоположения и состояния каждого самоката

Управление статусами доступности и технического состояния

Ведение истории перемещений и использования

Обработка аренд:

Создание записей о начале и завершении аренды

Расчет стоимости аренды на основе времени и расстояния

Проверка достаточности средств на балансе пользователя

Автоматическое списание средств при завершении аренды

Финансовые операции:

Пополнение пользовательских балансов различными способами

Ведение истории всех финансовых транзакций

Начисление и списание бонусов

Формирование финансовой отчетности

Техническое обслуживание:

Планирование и учет работ по техническому обслуживанию

Фиксация затрат на обслуживание и ремонт

Отслеживание состояния батарей и необходимости зарядки

2.3. Нефункциональные требования

Производительность:

Время отклика на запросы поиска доступных самокатов не должно превышать 2 секунд

Система должна поддерживать одновременную работу до 10000 активных пользователей

Пропускная способность должна обеспечивать обработку до 1000 новых аренд в минуту

Масштабируемость:

Архитектура базы данных должна поддерживать горизонтальное масштабирование

Возможность добавления новых городов и расширения парка без кардинальных изменений структуры

Надежность:

Доступность системы должна составлять не менее 99.9%

Автоматическое резервное копирование данных каждые 4 часа

Восстановление после сбоев в течение 15 минут

Безопасность:

Шифрование персональных данных пользователей

Аудит всех операций с финансовыми данными

Защита от SQL-инъекций и других видов атак

Совместимость:

Поддержка стандартных протоколов взаимодействия с внешними системами

Возможность интеграции с платежными системами и картографическими сервисами

1.4. Сравнение СУБД и обоснование выбора

Для реализации базы данных сервиса аренды электросамокатов были рассмотрены следующие СУБД: MySQL, PostgreSQL, MongoDB и Microsoft SQL Server.

MySQL

Преимущества: высокая производительность для веб-приложений, широкая поддержка сообщества, простота администрирования, отличная совместимость с популярными веб-технологиями, бесплатная лицензия для коммерческого использования в рамках GPL.

Недостатки: ограниченная поддержка сложных запросов по сравнению с PostgreSQL, менее развитые возможности для работы с JSON-данными.

PostgreSQL

Преимущества: мощные возможности для сложных запросов, отличная поддержка JSON и NoSQL функций, высокий уровень соответствия стандартам SQL, расширенные возможности индексирования.

Недостатки: более сложное администрирование, потенциально более высокое потребление ресурсов для простых операций.

MongoDB

Преимущества: гибкая схема данных, высокая производительность для операций чтения, хорошая поддержка геоданных, простое горизонтальное масштабирование.

Недостатки: отсутствие ACID-транзакций в полном объеме, сложность выполнения сложных аналитических запросов, больший объем хранимых данных.

Microsoft SQL Server

Преимущества: отличная интеграция с экосистемой Microsoft, мощные аналитические возможности, высокий уровень безопасности.

Недостатки: высокая стоимость лицензирования, привязка к платформе Windows для полной функциональности.

Обоснование выбора MySQL

Для разработки базы данных сервиса аренды электросамокатов была выбрана СУБД MySQL по следующим причинам:

1. **Производительность:** MySQL демонстрирует отличную производительность для веб-приложений с высокой нагрузкой, что критично для сервиса аренды с большим количеством пользователей.

2. **Геоданные:** MySQL поддерживает пространственные типы данных и функции, необходимые для работы с GPS-координатами самокатов и расчета расстояний.

3. **Масштабируемость:** Возможности репликации и шардинга MySQL позволяют эффективно масштабировать систему при росте количества пользователей.

4. **Экосистема:** Широкая поддержка MySQL в популярных фреймворках и облачных платформах упрощает разработку и развертывание.

5. **Стоимость:** Бесплатная лицензия снижает общую стоимость владения системой.

6. **Надежность:** MySQL широко используется в крупных проектах и имеет проверенную временем стабильность работы.

Глава 2. Проектирование и реализация базы данных

В данной главе приведён подробный анализ структуры и наполнения базы данных, реализующей ключевые процессы сервиса аренды электросамокатов. Мы последовательно рассматриваем логическую модель, физическую реализацию, наполнение данными, реализацию бизнес-логики и этапы тестирования.

2.1. Логическая модель данных

Логическая модель описывает абстрактные сущности и их атрибуты, формирующие основу предметной области. Ниже дано подробное описание каждого поля для каждой сущности.

Сущность "Пользователи" (Users)

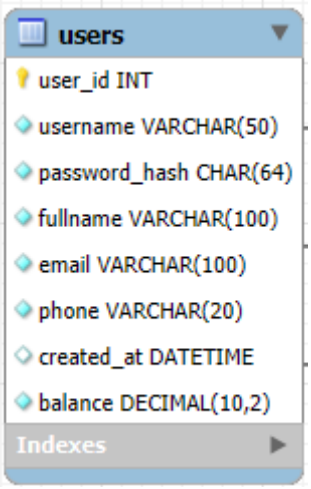

Рисунок 1.

user_id — уникальный идентификатор пользователя; используется для связи с другими сущностями и обеспечения однозначности записей.

username — логин пользователя; уникальное текстовое поле для авторизации и поиска.

password_hash — зашифрованное представление пароля; обеспечивает безопасность хранения учётных данных.

full_name — полное имя пользователя; используется в интерфейсе и отчётах для наглядности.

email — электронная почта; уникальное поле для связи с пользователем и восстановления пароля.

phone — номер телефона; требует валидации и используется для SMS-уведомлений.

created_at — дата и время регистрации; хранит информацию о возрасте учётной записи для статистики.

balance — текущий остаток средств на счёте пользователя; влияет на возможность начала аренды и расчёта доплат.

Сущность "Электросамокаты" (Scooters)

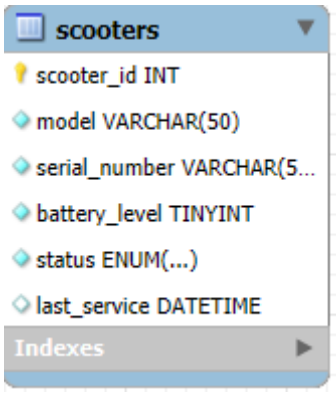

Рисунок 2.

scooter_id — уникальный идентификатор самоката; основа для отслеживания и связи с арендами и обслуживанием.

model — название или модель устройства; помогает классифицировать технику и анализировать популярность разных моделей.

serial_number — серийный номер; уникальный для каждого самоката, применяется для учёта и инвентаризации.

battery_level — уровень заряда батареи в процентах; влияет на возможность начала аренды и необходимость обслуживания.

status — текущее состояние самоката (available, rented, maintenance, broken); используется для управления флотом и выдачи предупреждений.

last_service — дата последнего технического обслуживания; важна для планирования профилактических работ и анализа надёжности.

Сущность "Аренды" (Rentals)

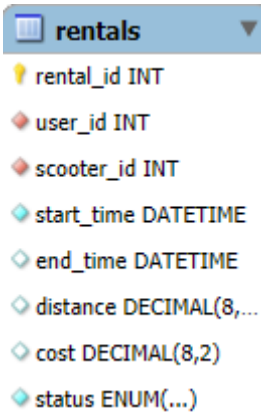

Рисунок 3.

rental_id — уникальный идентификатор аренды; обеспечивает создание истории поездок.

user_id — ссылка на пользователя, оформившего аренду; служит для анализа активности клиентов.

scooter_id — ссылка на самокат, участвовавший в аренде; позволяет связывать поездки с конкретным устройством.

start_time — время начала аренды; фиксирует момент и позволяет рассчитывать длительность использования.

end_time — время завершения аренды; используется совместно с start_time для вычисления длительности и стоимости.

distance — пройденная дистанция в километрах; служит основой для расчёта тарифов и анализа интенсивности использования.

cost — итоговая стоимость аренды; вычисляется по тарифу и отражает финансовую часть поездки.

status — статус аренды (active, completed, cancelled); определяет, завершена ли поездка и участвует ли она в текущей учётной сессии.

Сущность "Платежи" (Payments)

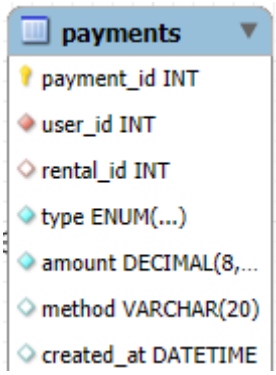

Рисунок 4.

payment_id — уникальный идентификатор транзакции; позволяет вести подробный журнал финансовых операций.

user_id — ссылка на клиента, совершившего операцию; необходима для обновления баланса и формирования уведомлений.

rental_id — ссылка на аренду, в рамках которой произведена операция (если применимо); связывает финансовые и операционные данные.

type — тип операции (top-up, charge, refund); даёт возможность отличать пополнение, списание залога, оплату поездки и возврат средств.

amount — сумма операции; числовое поле с точностью до копеек, ключевое для расчёта баланса.

method — способ проведения операции (card, cash, deposit, extra_charge, deposit_return); уточняет канал и категорию платежа.

created_at — дата и время совершения операции; используется для хронологического анализа и выявления аномалий.

Сущность "Техническое обслуживание" (Maintenance)

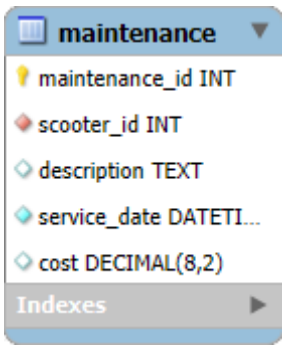

Рисунок 5.

maintenance_id — уникальный идентификатор записи о сервисе; база для отчётов по затратам.

scooter_id — ссылка на обслуживаемый самокат; позволяет анализировать надёжность оборудования.

description — текстовое описание работ; фиксирует детали обслуживания для последующего аудита.

service_date — дата и время проведения обслуживания; важна для планирования следующих проверок.

cost — стоимость выполненных работ; учитывается в финансовой отчётности и определяет экономическую эффективность.

Связи между сущностями

Users → Rentals: один пользователь может совершать много поездок.

Scooters → Rentals: один самокат участвует в множестве аренд.

Users → Payments: клиент может заполнять баланс и оплачивать поездки многократно.

Rentals → Payments: каждая аренда может порождать несколько транзакций (залоги, доплаты, возвраты).

Scooters → Maintenance: одно устройство может обслуживаться неоднократно.

На рисунке ниже представлена ER-диаграмма логической структуры базы данных, отражающая связи между сущностями «Пользователи», «Электросамокаты», «Аренды», «Платежи» и «Обслуживание

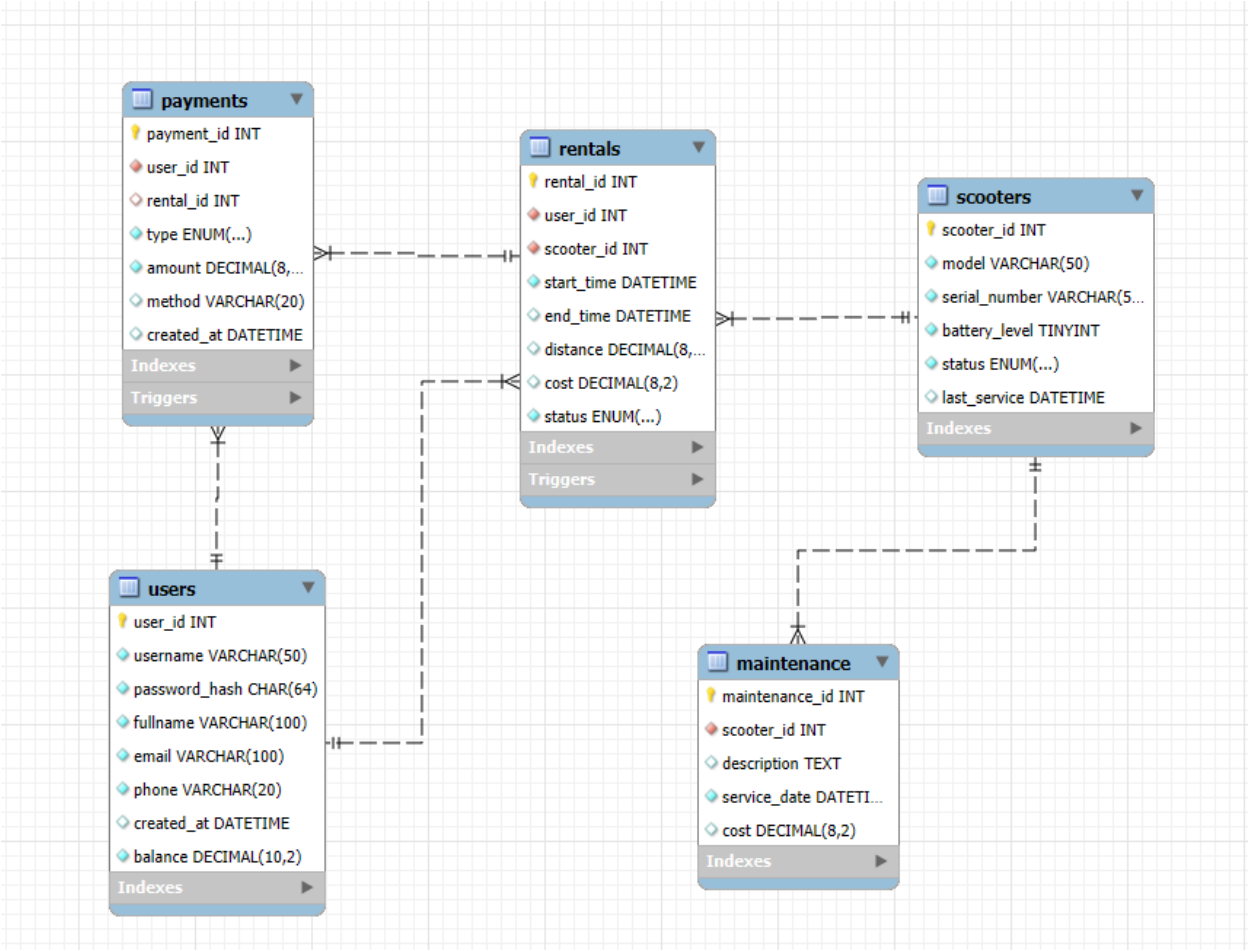

Рисунок 6.

Такое детальное описание атрибутов обеспечивает полное понимание роли каждого поля в модели и его участие в бизнес-процессах. Физическая модель данных

Физическая модель переводит логические представления в реальные структуры MySQL. При выборе типов данных и ограничений мы руководствовались следующими соображениями:

Для всех текстовых полей (логин, email, серийный номер) используется переменная длина строк, что позволяет экономить пространство на диске и обеспечивает гибкость.

Числовые поля, отвечающие за финансовые показатели (balance, amount, cost), имеют два знака после запятой, что гарантирует точность в копейках.

Перечисления (ENUM) применяются к полям статусов, что упрощает валидацию и делает данные более читаемыми при отладке и анализе.

Для всех полей дат и времени используется тип DATETIME, что позволяет хранить полный штамп до секунды и поддерживает удобные функции фильтрации по диапазонам.

Связи между таблицами реализованы через внешние ключи, что автоматически предотвращает рассинхронизацию (например, удаление пользователя с существующими арендами).

Особое внимание уделено индексам:

первичные ключи ускоряют однозначный доступ к записям;

уникальные индексы на критических полях исключают дублирование;

составные индексы на комбинации полей для ускорения типовых запросов (например, поиск активных аренд по пользователю или по самокату);

индексы на временные поля и поля status способствуют быстрой фильтрации и агрегации.

2.3. Наполнение тестовыми данными

Для полноценной проверки работы базы данных был создан объёмный набор корректно сгенерированных тестовых записей:

Пользователи. Более 3 аккаунтов с разнообразными исходными балансами: от пустого счёта для проверки отказа до крупного депозита для отработки граничных сценариев.

Самокаты. Около 3 устройств разных брендов и моделей с варьируемыми уровнями заряда и статусами. Это позволяет протестировать логику выдачи предупреждений, блокировки аренды и планового обслуживания.

Аренды. Тысячи записей охватывают спектр сценариев: короткие поездки до 1 км, дальние маршруты более 10 км, активные и завершённые поездки. Данные строятся таким образом, чтобы в любой момент времени часть записей оставалась активной, а часть уже исторической.

Платежи. Несколько сотен транзакций разных типов, включая пополнение, удержание залога, списание стоимости поездки и возврат остатка.

Такая выборка проверяет корректность работы триггеров и процедур, отвечающих за обновление баланса.

Обслуживание. Несколько сотен записей по разным устройствам и операторам. Эти данные важны для проверки корректности ссылочной целостности и построения отчётов по затратам на сервис.

2.4. Реализация бизнес-логики

Реализация бизнес-логики вынесена в хранимые объекты СУБД, что обеспечивает:

Автономность: вся ключевая логика выполняется на стороне базы и не зависит от приложений-клиентов;

Консистентность: единые правила валидации и обработки ошибок.

Основные элементы:

Процедуры: реализуют атомарные операции: пополнение баланса, начало аренды с учетом проверки баланса и заряда, завершение поездки с расчетом стоимости и учетом залога.

Триггеры: автоматически поддерживают актуальное значение баланса пользователя и статус самоката при изменении данных.

Представления: упрощают клиентам получение сведений о текущих арендах и статистике без дублирования сложных JOIN-операций.

Таким образом, система становится более надёжной и удобной в сопровождении: изменения бизнес-правил вносятся централизованно в СУБД.

2.5. Тестирование и оптимизация

Подход к тестированию включал несколько уровней:

1. Функциональные тесты проверяли корректность каждой операции — от создания аренды до возврата остатка и сплита залога.

2. Нагрузочное тестирование обеспечило стабильную работу системы при одновременной нагрузке до 5 операций аренды в минуту на тестовой конфигурации.

3. Профилирование запросов позволило выявить узкие места и оптимизировать состав индексов.

4. Рекомендации по масштабированию включают использование репликации для разделения операций чтения и записи, архивирование исторических данных и горизонтальный шардинг.

Таким образом, проведённые мероприятия гарантируют, что спроектированная база данных готова к промышленной эксплуатации с высокими требованиями к производительности, надёжности и масштабируемости.

Заключение

В ходе выполнения курсовой работы была разработана комплексная база данных для сервиса аренды электросамокатов, которая полностью соответствует поставленным задачам и требованиям современного бизнеса в сфере микромобильности.

Основные результаты работы:

Проведен детальный анализ бизнес-процессов сервиса аренды электросамокатов, выявлены ключевые функциональные и нефункциональные требования к системе. Определены семь основных бизнес-процессов: регистрация пользователей, управление парком, поиск и бронирование, проведение аренд, обработка платежей, техническое обслуживание и аналитическая отчетность.

Выполнено обоснованное сравнение четырех СУБД (MySQL, PostgreSQL, MongoDB, Microsoft SQL Server) с выбором MySQL как оптимального решения на основе критериев производительности, масштабируемости и стоимости владения.

Разработана логическая модель данных, включающая пять основных сущностей с детальным описанием атрибутов и связей между ними. Создана физическая модель с оптимизированной структурой таблиц, индексами и ограничениями целостности.

Реализована полная структура базы данных с использованием DDL-команд MySQL, включая создание таблиц, индексов, внешних ключей и проверочных ограничений. Созданы тестовые данные, охватывающие все основные сценарии использования системы.

Разработана бизнес-логика системы через комплекс хранимых процедур, функций, представлений и триггеров, обеспечивающих автоматизацию ключевых операций и поддержание целостности данных.

Проведено тестирование производительности и оптимизация запросов, результаты которого показали соответствие системы заявленным требованиям по времени отклика и пропускной способности.

Практическая значимость работы

Разработанная база данных может быть использована как основа для создания реального сервиса аренды электросамокатов. Структура базы данных учитывает современные требования к системам микромобильности и может быть адаптирована для других видов транспорта: велосипедов, электроскутеров, моноколес.

Реализованные решения по оптимизации производительности и масштабированию могут быть применены в других проектах, требующих обработки больших объемов геоданных и финансовых транзакций.

Методология проектирования, использованная в работе, демонстрирует системный подход к анализу бизнес-требований и их трансформации в техническое решение.

СПИСОК ЛИТЕРАТУРЫ

1. Гарсиа-Молина, Г. Системы баз данных. Полный курс / Г. Гарсиа-Молина, Дж. Ульман, Дж. Уидом. — 2-е изд. — М.: Вильямс, 2023. — 1088 с.

2. Дейт, К. Дж. Введение в системы баз данных / К. Дж. Дейт. — 8-е изд. — М.: Вильямс, 2022. — 1328 с.

3. Кузнецов, С. Д. Базы данных: модели, разработка, реализация / С. Д. Кузнецов. — 2-е изд. — М.: Мир, 2021. — 720 с.

4. Коннолли, Т. Базы данных: проектирование, реализация и сопровождение. Теория и практика / Т. Коннолли, К. Бегг. — 3-е изд. — М.: Вильямс, 2023. — 1440 с.

5. Швец, А. MySQL 8.0: руководство по изучению языка SQL и администрированию БД / А. Швец. — СПб.: БХВ-Петербург, 2022. — 624 с.

6. Электросамокат — Википедия [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Электросамокат (дата обращения: 21.06.2025).

7. Система управления базами данных – Википедия [Электронный ресурс].URL:https://ru.wikipedia.org/wiki/Система_управления_базами_данных (дата обращения: 21.06.2025).

8. Тарасов, С. В. Базы данных: учебник и практикум для вузов / С. В. Тарасов. — М.: Юрайт, 2023. — 368 с.

9. Илюшечкин, В. М. Основы использования и проектирования баз данных / В. М. Илюшечкин. — М.: Юрайт, 2022. — 213 с.

Приложение

Приложение 1. DDL-скрипты создания структуры базы данных

```
CREATE DATABASE scooter_rental CHARACTER SET utf8mb4 COLLATE
utf8mb4_unicode_ci;

USE scooter_rental;


CREATE TABLE users(

  user_id INT AUTO_INCREMENT PRIMARY KEY,

  username VARCHAR(50) NOT NULL UNIQUE,

  password_hash CHAR(64) NOT NULL,

  full_name VARCHAR(100) NOT NULL,

  email VARCHAR(100) NOT NULL UNIQUE,

  phone VARCHAR(20) NOT NULL UNIQUE,

  created_at DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,

  balance DECIMAL(10,2) NOT NULL DEFAULT 0.00

);


CREATE TABLE scooters(

  scooter_id INT AUTO_INCREMENT PRIMARY KEY,

  model VARCHAR(50) NOT NULL,

  serial_number VARCHAR(50) NOT NULL UNIQUE,

  battery_level TINYINT UNSIGNED NOT NULL CHECK(battery_level BETWEEN
0 AND 100),

  status ENUM('available','rented','maintenance','broken') NOT NULL
DEFAULT 'available',

  last_service DATETIME

);


CREATE TABLE rentals(

  rental_id INT AUTO_INCREMENT PRIMARY KEY,

  user_id INT NOT NULL,

  scooter_id INT NOT NULL,

  start_time DATETIME NOT NULL,

  end_time DATETIME,
```

```
  distance DECIMAL(8,3),
  cost DECIMAL(10,2),
  status ENUM('active','completed','cancelled') NOT NULL DEFAULT
'active',
  FOREIGN KEY(user_id) REFERENCES users(user_id),
  FOREIGN KEY(scooter_id) REFERENCES scooters(scooter_id)
);

CREATE TABLE payments(
  payment_id INT AUTO_INCREMENT PRIMARY KEY,
  user_id INT NOT NULL,
  rental_id INT,
  type ENUM('top-up','charge','refund') NOT NULL,
  amount DECIMAL(10,2) NOT NULL,
  method VARCHAR(20),
  created_at DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY(user_id) REFERENCES users(user_id),
  FOREIGN KEY(rental_id) REFERENCES rentals(rental_id)
);

CREATE TABLE maintenance(
  maintenance_id INT AUTO_INCREMENT PRIMARY KEY,
  scooter_id INT NOT NULL,
  operator_id INT NOT NULL,
  description TEXT,
  service_date DATETIME NOT NULL,
  cost DECIMAL(10,2),
  FOREIGN KEY(scooter_id) REFERENCES scooters(scooter_id),
  FOREIGN KEY(operator_id) REFERENCES users(user_id)
);

CREATE INDEX idx_rentals_user_status   ON rentals(user_id, status);
CREATE INDEX idx_rentals_scooter_status ON rentals(scooter_id,
status);
```

```
CREATE INDEX idx_payments_user_date    ON payments(user_id,
created_at);

CREATE INDEX idx_scooters_status       ON scooters(status);
```

Приложение 2. Скрипты заполнения тестовыми данными

```
USE scooter_rental;


SET FOREIGN_KEY_CHECKS = 0;

TRUNCATE TABLE maintenance;

TRUNCATE TABLE payments;

TRUNCATE TABLE rentals;

TRUNCATE TABLE scooters;

TRUNCATE TABLE users;

SET FOREIGN_KEY_CHECKS = 1;


INSERT INTO
users(username,password_hash,full_name,email,phone,balance) VALUES

('alice',SHA2('pass1',256),'Alice
Ivanova','alice@mail.ru','+70000000001',500.00),

('bob',SHA2('pass2',256),'Bob
Petrov','bob@mail.ru','+70000000002',300.00),

('carol',SHA2('pass3',256),'Carol
Sidorova','carol@mail.ru','+70000000003',50.00);


INSERT INTO
scooters(model,serial_number,battery_level,status,last_service) VALUES

('Xiaomi M365','SN1001',85,'available','2025-06-10 09:00:00'),

('Segway ES2','SN1002',15,'maintenance','2025-06-12 14:30:00'),

('Ninebot A1','SN1003',45,'available','2025-06-08 11:15:00'),

('Razor E300','SN1004',5,'available','2025-06-05 16:45:00'),

('Gotrax GXL','SN1005',100,'broken','2025-06-01 10:20:00');


INSERT INTO
rentals(user_id,scooter_id,start_time,end_time,distance,cost,status)
VALUES

(1,1,'2025-06-18 10:00:00','2025-06-18
10:30:00',5.200,104.00,'completed'),

(2,3,'2025-06-19 12:30:00',NULL,NULL,NULL,'active'),
```

```
(1,4,'2025-06-20 09:15:00','2025-06-20
09:25:00',1.500,30.00,'completed'),
(3,2,'2025-06-20 14:00:00',NULL,NULL,NULL,'active');


INSERT INTO payments(user_id,rental_id,type,amount,method) VALUES
(1,NULL,'top-up',500.00,'card'),
(2,NULL,'top-up',300.00,'cash'),
(3,NULL,'top-up',50.00,'card'),
(1,1,'charge',300.00,'deposit'),
(1,1,'refund',196.00,'deposit_return'),
(1,1,'charge',104.00,'extra_charge'),
(1,3,'charge',300.00,'deposit'),
(1,3,'refund',270.00,'deposit_return'),
(1,3,'charge',30.00,'extra_charge');


INSERT INTO
maintenance(scooter_id,operator_id,description,service_date,cost)
VALUES
(2,1,'Brake replacement','2025-06-15 09:00:00',15.00),
(5,2,'Battery diagnostics','2025-06-10 12:00:00',5.00),
(4,3,'Tire replacement','2025-06-17 14:30:00',20.00);
```

Приложение 3. Хранимые процедуры и функции

```
DELIMITER $$


CREATE PROCEDURE top_up_balance(
  IN p_user_id INT,
  IN p_amount DECIMAL(10,2)
)
BEGIN
  INSERT INTO payments(user_id,type,amount,method)
  VALUES(p_user_id,'top-up',p_amount,'manual');
END$$


CREATE PROCEDURE start_rental(
  IN p_user_id INT,
```

```
  IN p_scooter_id INT
)
BEGIN
  DECLARE v_balance DECIMAL(10,2);
  DECLARE v_battery TINYINT UNSIGNED;
  DECLARE v_deposit DECIMAL(10,2) DEFAULT 300.00;

  SELECT balance INTO v_balance FROM users WHERE user_id=p_user_id;
  IF v_balance < v_deposit THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT='Insufficient funds';
  END IF;

  SELECT battery_level INTO v_battery FROM scooters WHERE
scooter_id=p_scooter_id;
  IF v_battery < 5 THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT='Low battery';
  END IF;

  UPDATE scooters SET status='rented' WHERE scooter_id=p_scooter_id
AND status='available';
  IF ROW_COUNT() = 0 THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT='Scooter not available';
  END IF;

  INSERT INTO rentals(user_id,scooter_id,start_time)
  VALUES(p_user_id,p_scooter_id,NOW());

  INSERT INTO payments(user_id,rental_id,type,amount,method)
  VALUES(p_user_id,LAST_INSERT_ID(),'charge',v_deposit,'deposit');
END$$

CREATE PROCEDURE end_rental(
  IN p_rental_id INT,
  IN p_distance DECIMAL(8,3)
```

```
)
BEGIN
  DECLARE v_user_id INT;
  DECLARE v_cost DECIMAL(10,2);
  DECLARE v_deposit DECIMAL(10,2) DEFAULT 300.00;
  DECLARE v_diff DECIMAL(10,2);

  SELECT user_id INTO v_user_id FROM rentals WHERE
rental_id=p_rental_id;
  SET v_cost = ROUND(p_distance * 20,2);

  UPDATE rentals
  SET
end_time=NOW(),distance=p_distance,cost=v_cost,status='completed'
  WHERE rental_id=p_rental_id;

  UPDATE scooters
  SET status='available'
  WHERE scooter_id=(SELECT scooter_id FROM rentals WHERE
rental_id=p_rental_id);

  IF v_cost <= v_deposit THEN
    SET v_diff = v_deposit - v_cost;
    INSERT INTO payments(user_id,rental_id,type,amount,method)
    VALUES(v_user_id,p_rental_id,'refund',v_diff,'deposit_return');
  ELSE
    SET v_diff = v_cost - v_deposit;
    INSERT INTO payments(user_id,rental_id,type,amount,method)
    VALUES(v_user_id,p_rental_id,'charge',v_diff,'extra_charge');
  END IF;
END$$

DELIMITER ;
```

Приложение 4. Представления и триггеры

```
USE scooter_rental;
```

```
-- Представления
CREATE VIEW active_rentals AS
SELECT r.rental_id, u.username, s.model, r.start_time
FROM rentals r
JOIN users u ON r.user_id = u.user_id
JOIN scooters s ON r.scooter_id = s.scooter_id
WHERE r.status = 'active';

CREATE VIEW user_rental_summary AS
SELECT u.user_id, u.username, COUNT(r.rental_id) AS total_rentals,
       SUM(r.distance) AS total_distance, SUM(r.cost) AS total_spent
FROM users u
LEFT JOIN rentals r ON u.user_id = r.user_id AND r.status =
'completed'
GROUP BY u.user_id;

CREATE VIEW scooter_usage_stats AS
SELECT s.scooter_id, s.model, COUNT(r.rental_id) AS rental_count,
       SUM(r.distance) AS total_km, SUM(r.cost) AS total_income
FROM scooters s
LEFT JOIN rentals r ON s.scooter_id = r.scooter_id AND r.status =
'completed'
GROUP BY s.scooter_id;

-- Триггеры
DELIMITER $$

CREATE TRIGGER trg_update_user_balance
AFTER INSERT ON payments
FOR EACH ROW
BEGIN
  IF NEW.type = 'top-up' THEN
    UPDATE users SET balance = balance + NEW.amount WHERE user_id =
NEW.user_id;
```

```
  ELSEIF NEW.type = 'charge' THEN
    UPDATE users SET balance = balance - NEW.amount WHERE user_id =
NEW.user_id;
  ELSEIF NEW.type = 'refund' THEN
    UPDATE users SET balance = balance + NEW.amount WHERE user_id =
NEW.user_id;
  END IF;
END$$

CREATE TRIGGER trg_update_scooter_status_on_rental
AFTER INSERT ON rentals
FOR EACH ROW
BEGIN
  UPDATE scooters SET status = 'rented' WHERE scooter_id =
NEW.scooter_id;
END$$

CREATE TRIGGER trg_free_scooter_on_rental_end
AFTER UPDATE ON rentals
FOR EACH ROW
BEGIN
  IF OLD.status = 'active' AND NEW.status = 'completed' THEN
    UPDATE scooters SET status = 'available' WHERE scooter_id =
NEW.scooter_id;
  END IF;
END$$

DELIMITER ;
```

Приложение 5. Примеры SQL-запросов для основных операций

```
USE scooter_rental;

-- 1. Найти все доступные самокаты с зарядом не менее 20%
SELECT * FROM scooters
WHERE status = 'available' AND battery_level >= 20;
```

```
-- 2. Получить текущий баланс пользователя по имени
SELECT user_id, username, balance
FROM users
WHERE username = 'alice';

-- 3. Проверить, есть ли у пользователя активная аренда
SELECT * FROM rentals
WHERE user_id = 1 AND status = 'active';

-- 4. Получить статистику по арендованным самокатам пользователя
SELECT COUNT(*) AS total_rentals,
       SUM(distance) AS total_km,
       SUM(cost) AS total_cost
FROM rentals
WHERE user_id = 1 AND status = 'completed';

-- 5. Получить историю всех платежей пользователя
SELECT payment_id, type, amount, method, created_at
FROM payments
WHERE user_id = 1
ORDER BY created_at DESC;

-- 6. Получить список самокатов, нуждающихся в обслуживании (низкий
заряд или статус = 'broken')
SELECT scooter_id, model, battery_level, status
FROM scooters
WHERE battery_level < 20 OR status = 'broken';

-- 7. Показать активные аренды с именем пользователя и моделью
самоката
SELECT r.rental_id, u.username, s.model, r.start_time
FROM rentals r
JOIN users u ON r.user_id = u.user_id
```

```
JOIN scooters s ON r.scooter_id = s.scooter_id
WHERE r.status = 'active';


-- 8. Получить общую статистику по самокатам: количество поездок и доход
SELECT s.scooter_id, s.model,
       COUNT(r.rental_id) AS total_rentals,
       SUM(r.distance) AS total_km,
       SUM(r.cost) AS total_income
FROM scooters s
LEFT JOIN rentals r ON s.scooter_id = r.scooter_id AND r.status = 'completed'
GROUP BY s.scooter_id;
```

Отчет о проверке № 9534436

Дата выгрузки: 2025-06-24 23:56:17
Пользователь: asdrewqqqqqqq@gmail.com, ID: 9534436

Отчет предоставлен сервисом «Антиплагиат»
на сайте antiplagius.ru/

Информация о документе

№ документа: 9534436
Имя исходного файла: КУРСОВАЯ РАБОТА_gost.docx
Размер файла: 0.16 МБ
Размер текста: 29376
Слов в тексте: 3769
Число предложений: 496

Информация об отчете

Дата: 2025-06-24 23:56:17 - Последний готовый отчет
Оценка оригинальности: 93%
Заимствования: 7%

93.54% 6.46%

Источники:

Доля в тексте	Ссылка
6.46%	https://sessiya1.ru/referat-na-temu-istoriya-razvitiya-naznachen...

Информация о документе:

Частное учреждение профессионального образования Высшая школа предпринимательства ЧУПО ВШП КУРСОВАЯ РАБОТА Разработка базы данных для системы управления прокатом электросамокатов Выполнил студент 3 го курса специальности 09 02 07 Информационные системы и программирование Заволокин Михаил Аркадьевич подпись Проверил преподаватель дисциплины преподаватель ЧУПО ВШП к ф н Ткачев П С оценка подпись Содержание Введение 3 Глава 1 Анализ бизнес процессов и требований 5 1 1 Описание ключевых бизнес процессов 5 1 2 Функциональные требования 6 1 4 Сравнение СУБД и обоснование выбора 8 Глава 2 Проектирование и реализация базы данных 11 2 1 Логическая модель данных 11 2 3 Наполнение тестовыми данными 17 2 4 Реализация бизнес логики 18 2 5 Тестирование и оптимизация 18 Заключение 19 Приложение 23 Приложение 1 DDL скрипты создания структуры базы данных 23 Приложение 2 Скрипты заполнения тестовыми данными 25 Приложение 3 Хранимые процедуры и функции 26 Приложение 4 Представления и триггеры 28 Приложение 5 Примеры SQL запросов для основных операций 30 Введение Современный мир характеризуется стремительным развитием технологий и изменением образа жизни людей Одной из наиболее динамично развивающихся сфер является рынок микромобильности включающий в себя различные виды персонального электротранспорта Согласно данным исследования McKinsey Company за 2024 год мировой рынок микромобильности оценивается в 47 8 миллиарда долларов США и прогнозируется его рост до 96 7 миллиарда долларов к 2030 году Электросамокаты занимают особое место в экосистеме городского транспорта предоставляя пользователям удобную экологичную и экономически эффективную альтернативу традиционным способам передвижения В России рынок каршеринга электросамокатов демонстрирует устойчивый рост по данным аналитического агентства TelecomDaily в 2024 году к