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

КУРСОВАЯ РАБОТА

«Разработка базы данных для системы управления арендой автомобилей»

Выполнил:
студент 3-го курса специальности
09.02.07 «Информационные системы
и программирование»
Дедюхин Денис Алексеевич
подпись: _____

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

Тверь, 2025 г.

Содержание

Введение

Актуальность

Сейчас автомобили все чаще становиться не роскошью, а необходимостью, особенно в условиях динамичного ритма жизни. Однако не каждый человек может позволить себе покупку собственного авто по финансовым или другим причинам. Поэтому все больше людей прибегают к услуге аренды автомобилей. Эта тема считается актуальной, так как позволяет людям решать транспортные задачи без крупных вложений. Кроме того, многие процессы, используемых в сфере аренды автомобилей, могут быть адаптированы и применены в другие бизнес - направлениях или сервисных концепциях.

Определение цели работы

Цель работы - разработать базу данных, отражающую основные бизнес-процессы, характерные для условной компании, предоставляющей услуги аренды авто.

Прежде чем перейти к формулировке задач, необходимо для достижения поставленной цели, стоит кратко изложить, что из себя вообще представляет «сервис аренды автомобилей» как сущность или явление.

Сервис аренды автомобилей - это услуга, предоставляющая клиентам возможность временного пользования транспортным средством за определённую плату. Такой подход удобен для тех, кто не обладает личным автомобилем, либо нуждается в нем лишь на короткий срок - например, для поездок в другой город, отдых или временной замены. Бизнес в этой сфере включает в себя множество процессов: ведение учёта автомобилей, клиентов, менеджеров, расчёт стоимости, отслеживание статусов авто, а также контроль возврата автомобилей и их техническое состояние.

Разработка базы данных позволяет автоматизировать и систематизировать эти процессы, обеспечить надёжность хранения информации, удобство взаимодействия между участниками системы и точность в расчётах.

Краткая история возникновения феномена в России.

Аренда автомобилей как услуга появилась в России в 1990-х голах с началом рыночных реформ. Первоначально она была ориентирована на иностранных туристов и бизнесменов, а доступ к ней имел лишь немногие из-за высокой стоимости

С развитием экономики и цифровых технологий в 2000-х годах начался рост интереса к краткосрочной аренде. Появились первые мобильные приложения, онлайн-бронирования, системы учёта и аналитики.К 2010-м годам аренда автомобилей стала массовым явлением в крупных городах.

Сегодня аренда авто - это востребованная слуга, позволяющая временно пользоваться транспортом без необходимости владения. Это стало возможным благодаря развитию ИТ-сервисов и автоматизированных баз данных, обеспечивающих контроль, удобство и прозрачность всех процессов.

Глава 1.

Постановка задач

Исходя из определения выше - сформулируем возможные задачи для достижения цели курсовой работы:

1. Определить основные бизнес - процессы условного сервиса аренды автомобилей.
2. Сформулировать требования к разрабатываемой базе данных с учётом специфики предметной области.
3. Выбрать СУБД (систему управления базами данных) для реализации базы данных.
4. Построить логическую и физическую схему базы данных.
5. Разработать набор взаимосвязанных таблиц, отражающих бизнес - процессы компании по аренде автомобилей.
6. Заполнить таблицы тестовыми данными для последующего тестирования.
7. Реализовать ключевые бизнес - процессы по уровне базы данных средствами выбранной СУБД (например, с помощью триггеров, процедур, представлений, функций и ролей.).
8. Провести тестирование реализованных механизмов с использованием тестовых данных, проанализировать корректность и полноту работы системы.

Объект исследования

Объектом исследования является процесс разработки базы данных, предназначенной для обслуживания бизнес - процессов условного сервиса аренды автомобилей.

Метод исследования

В работе используется подход к проектированию базы данных, а также методы структурного анализа. Реализация осуществляется с помощью языка SQL и инструментами выбранной СУБД.

Определить основные бизнес - процессы условного сервиса аренды автомобилей.

На первом этапе проектирование информационной системы необходимо определить основные бизнес-процессы, характерный для компании, предоставляющие услугу аренды автомобилей. Это является фундаментом всей системы, поскольку от правильного понимания и описания этих процессов зависит точность проектируемой модели базы данных, её структуры и логике функционирования.

Бизнес-процессы отражают реальные действия, которые происходят внутри организации: от момента первого взаимодействия с клиентом до завершения сделки аренды и анализа полученных данных. Каждому такому процессу соответствуют определённые сущности (например, клиент, автомобиль, менеджер, договор аренды), а также взаимосвязи между ними (например, один клиент может заключить несколько аренд, каждая аренда привязана к конкретному автомобилю и сотруднику)

Определение бизнес-процессов позволяет понять, какую информацию необходимо фиксировать, какие поля включать в таблицы, какие ограничения применять, и какие операции выполнять автоматически (например, расчёт стоимости аренды или изменения статуса автомобиля). Также на этом этапе выявляется потенциальные сценарии ошибок, риски и узкие места в работе сервиса, что даёт возможность заложить соответствующие механизмы контроля и владения в базе данных.

Таким образом, детальное описание и анализ бизнес-процессов - важный аналитический этап, который обеспечивает надёжную основу для построения логической модели базы данных, разработки удобного интерфейса и обеспечения корректной работы всей системы в целом.

К ключевым бизнес - процессам можно отнести:

- Регистрация клиентов, желающих арендовать автомобиль.
- Учёт автомобилей, включая их состояние, пробег, стоимость аренды и текущий статус (доступен, арендован, на обслуживании).
- Оформление аренды автомобиля, включая выбор машины, менеджера, сроков аренды и расчёт итоговой стоимости.
- Возврат автомобиля, обновление статуса и пробега, а также перерасчёт суммы аренды при изменении сроков.
- Работа менеджеров, сопровождающих клиентов и оформляющих автомобилей.
- Ведение журнала изменений для отслеживания истории статусов автомобилей.

Понимание этих процессов позволяет более точно построить структуру базы данных и реализовать логину её работа, ориентируясь на реальные задачи сервиса аренды.

Сформулировать требования к разрабатываемой базе данных с учётом специфики предметной области.

Для создания эффективной и надёжной информационной системы, обслуживающей сервис аренды автомобилей, необходимо заранее сформулировать требования к структуре и функциональности базы данных. Эти требования должны соответствовать особенностям предметной области и учитывать бизнес-логику, характерную для данной сферы.

1. Хранение ключевых данных.

Система должна хранить данные о клиентах, автомобилях, менеджерах и арендах.

- Для клиентов необходимо сохранять ФИО, контактные данные и статус (например, активен или находиться в чёрном списке).
- Для автомобилей - уникальные идентификатор (VIN), модель, пробег, стоимость аренды за сутки и текущий статус (доступен, арендован, на обслуживании)
- Для менеджеров - данные об обслуживающем персонале, включая имя, контактную информацию и процент комиссии.
- Для аренды - данные о дате начала и окончание аренды, участниках сделки, связанном автомобиле и общей стоимости.

2. Поддержка операций с данными.

База данных должна позволять выполнять все основные операции: добавление, редактирование и удаление записей. Однако эти действия должны быть строго ограничены бизнес-логикой, чтобы исключить появление противоречивой или некорректной информации. Например:

- Нельзя удалять автомобиль если он участвует в активной аренде.
- Нельзя добавлять аренду, если клиент находиться в чёрном списке или автомобиль временно недоступен.

3. Контроль доступности автомобиля.

Система должна проверять наличие свободных автомобилей на момент оформления аренды. Это требование реализуется с помощью проверки текущего статуса автомобиля (например, available, rented, maintenance) и исключает возможность двойного бронирования.

4. Автоматический расчёт стоимости аренды.

Важной частью бизнес-логики является автоматический расчёт общей стоимости аренды. Стоимость должна вычисляться на основании количества дней аренды и установленной суточной ставки для выбранного автомобиля. Это не только ускоряет процесс оформления, но и снижает риск ошибок при ручных расчётах.

5. Учёт изменений статуса автомобиля.

В системе должна быть реализована возможность фиксировать изменения статуса автомобиля от “доступен” до “арендован” и обратно, а также отслеживание технического состояния парка и планирования доступной машины.

6. Легирование операций и изменений.

Для повышения прозрачности и возможности анализа действий пользователей, необходимо предусмотреть ведение журнала изменений. Легирование можно фиксировать:

- Дату и время события.
- Измеренные значения.
- Идентификатор пользователя или процедуры, вызвавшей изменение.

Выбрать СУБД (систему управления базами данных) для реализации базы данных.

Для реализации базы данных было выбрано программное обеспечение MySQL. Данная система управления базами данных является одно из самых популярных и широко применяемых в мире благодаря своей надёжности, высокой производительности и открытой лицензии.

Выбор MySQL обусловлен следующими причинами:

- Поддержка стандартного языка SQL, что обеспечивает простоту разработки и возможность переноса решений из других СУБД при необходимости.
- Хорошая документация и активное сообщество, что упрощает процесс поиска решений при возникновении ошибок или нестандартных ситуаций.
- Поддержка транзакций, хранимых процедур, триггеров и пользовательских функций, что необходимо для реализации бизнес-логики внутри базы данных.
- Надёжность и устойчивость к сбоям, особенно при работе с большими объёмами данных.
- Широкая совместимость с различными средами разработки и интерфейсами, такими как PHP, Python, Java.

Таким образом, MySQL полностью удовлетворяет требованиям, предъявляемым к проектируемой базе данных, и позволяет эффективно реализовать все поставленные задача.

Построить логическую и физическую схему базы данных.

Процесс проектирования базы данных начинается с построение логической схемы, в которой определяются основные сущности предметной области, их атрибуты и взаимосвязи между ними. Логическая модель не зависит от конкретной СУБД и отражает общее представление о структуру данных.

В рамках условного сервиса аренды автомобилей были выделены следующие ключевые сущности:

- Клиенты - лица, оформляющие аренду автомобилей.
- Автомобили - транспортные средства, доступные для аренды.
- Менеджеры - сотрудники, сопровождающие процесс аренды.
- Аренды - записи о фактах аренды автомобилей клиентами.
- Журнал изменений - хранит информацию об изменениях статуса автомобилей.

Каждая сущность обладает набором атрибутов. Например, для автомобилей - это VIN, модель, пробег, стоимость аренды и текущий статус. Для аренды - даты начала и окончания, арендованная машина, клиент и менеджер, а также итоговая сумма аренды.

При проектирование базы данных для системы аренды автомобилей была выбрана нормализация до третьей нормальной формы (3NF). Это позволит добиться логической непротиворечивости, устранение избыточности и обеспечивать устойчивость структуры данных к изменениям.

После построения логической схемы разрабатывается физическая схема (ER-диаграмма), в которой логические сущности реализуются в виде таблиц с определёнными типами данных, ограничениями (например NOT NULL, UNIQUE, FOREGN KEY) и связями между ними (один-к-одному, один-ко-многим и т.д).

Важно отметить, что логическая модель не привязана к конкретной системе управления базами данных. Она создаётся на абстрактном уровне и служит универсальной основной для дальнейшего этапа - построения физической модели,

уже с учётом конкретных технических ограничений и особенностей выбранной СУБД.

Таким образом, построение логической схемы является критическим этапом проектирования, который позволяет тщательно проанализировать структуру будущей базы данных, обеспечить её корректность, логическую целостность и подготовить основу для реализации эффективной, масштабируемой и функциональной информационной системы.

Глава 2.

Разработать набор взаимосвязанных таблиц, отражающих бизнес - процессы компании по аренде автомобилей.

В рамках разработки информационной системы для условного сервиса аренды автомобилей был создан набор взаимосвязанных таблиц, каждая из которых отражает определённый аспект бизнес-процессов. Структуру таблиц представлена на ER-диаграмме (Рис.1.) и охватывает ключевые сущности предметной области.

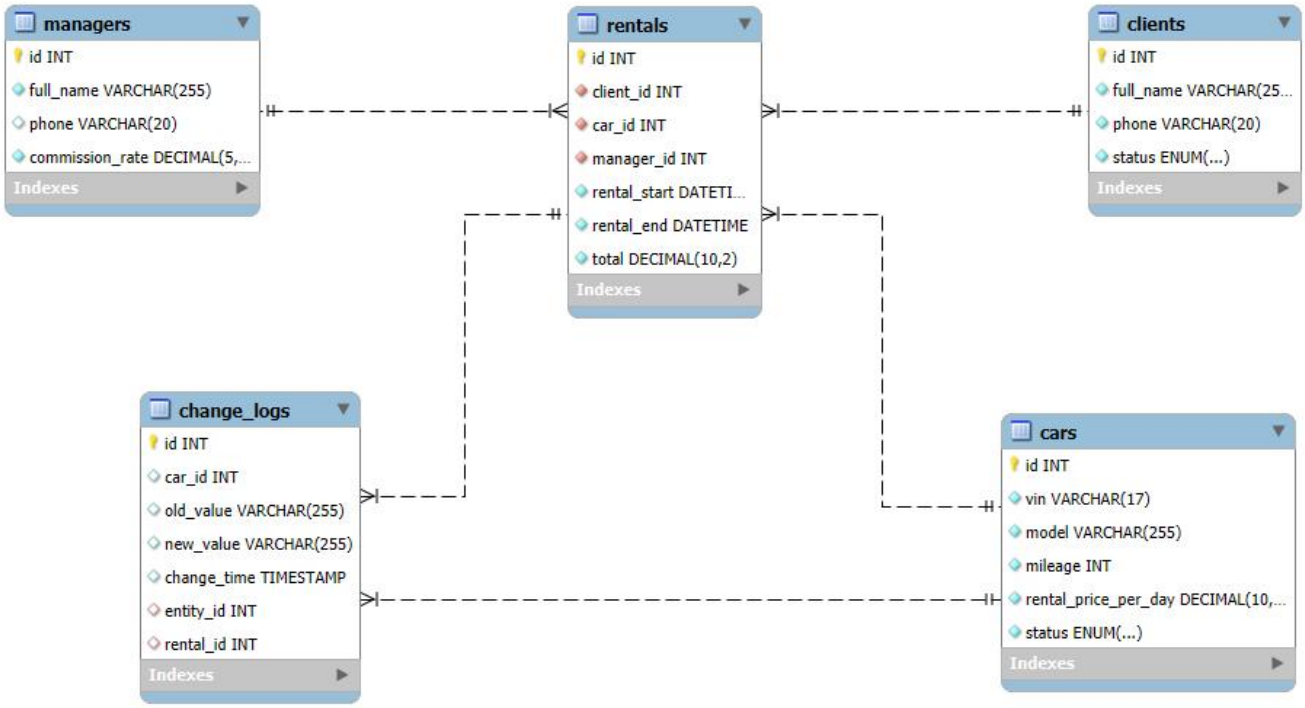

Рис.1.

Таблица cars содержит поля: id, vin, model, mileage, rental_price_per_day, status.

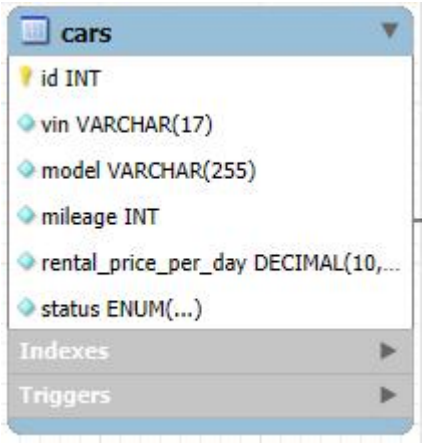

Рис.2.

- Для поля id выбран тип данных INT, поскольку это первичный ключ таблицы.
- Он используется для уникально идентификации записи и является автоинкрементируемым.
- Для поля vin выбран тип данных VARCHAR(17), так как VIN-номер автомобиля состоит из 17 символов, включая как буквы, так и цифры. Тип VARCHAR позволяеть гибко хранить такой формат.
- Для поля model выбран тип Varchar(255), поскольку название моделей автомобилей могут быть различные длины и включать как латинские, так и руские символы, а также пробелы и цифры.
- Для mileage используется тип INT, так как пробег автомобиля измеряется в километрах и всегда представляется в виде целого числа.
- Для поля rental_price_per_day используется тип DECIMAL(10,2), потому что это поле хранит денежное значение с точностью до двух знаков после запятой, что важно для расчётов стоимости аренды.
- Поле status реализован с использованием типа ENUM, в который входят значения ‘available’, ‘rented’, ‘maintenance’. Такой подход позволяет ограничить возможные статусы автомобиля только заранее определёнными значениями, минимизируя ошибки и повышая целостность данных.

Таблица clients содержит поля: id, full_name, phone, status

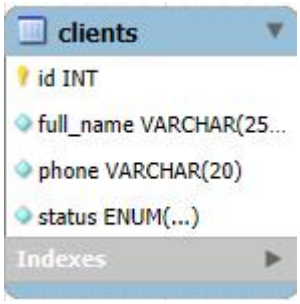

Рис.3.

- Для поля id выбран тип данных INT, так как оно ялвяеться ключем и должно быть автоинкрементируемым для уникальной идентификации клиента.
- Для поля full_name используется тип VARCHAR(255), так как ФИО клиента может содержать как кириллические, так и латинские символы, пробелы и иметь различную длину.
- Для поля phone выбран тип VARHCAR(255), поскольку номер телефона может содержать символы < - >, < + >, пробелы и код страны, что бы требует хранения как строки.
- Для поля status выбран тип ENUM(‘active’, ‘blacklisted’), чтобы ограничить возможные значения статуса клиента только допустимыми вариантами.

Таблица managers содержит поля: id, full_name, phone, commission_rate

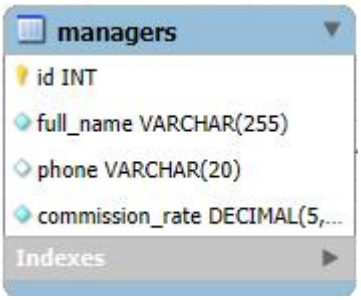

Рис.4.

- Поле id имеет тип INT - первичный ключ, автоинкрементируемый.
- Для поля full_name выбран тип VARCHAR(255) - для хранения полного имени менеджера в свободной форме.
- Поле phone имеет тип VARCHAR(20), как и у клиентов, для хранения телефонного номера с символами

- Для поля commission_rate используется DECIMAL(5, 2), так как процент комиссии может быть дробным (например, 0.15 = 15%). Это позволяет точно хранить значения до двух знаков после запятой.

Таблица rentals содержит поля: id, client_id, car_id, manager_id, rental_start, rental_end, total.

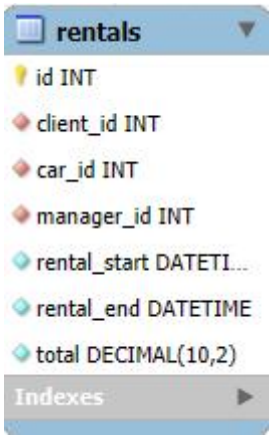

Рис.5.

- Поле id - тип INT, автоинкрементируемый первичный ключ.
- Поле client_id, car_id, manager_id - тип INT, так как они являются внешними ключами, ссылающимися на соответсвующие таблицы.
- Поля rental_start, rental_end - тип DATETIME, поскольку необходимо хранить точные дату и время начало и окончания аренды.
- Поле total имеет тип DECIMAL(10, 2) для хранения общей суммы аренды с двумя знаками после запятой. Что особенно важно при денежных расчётов.

Таблица change_logs содержит поля: id, entity_id, old_value, new_value, change_time

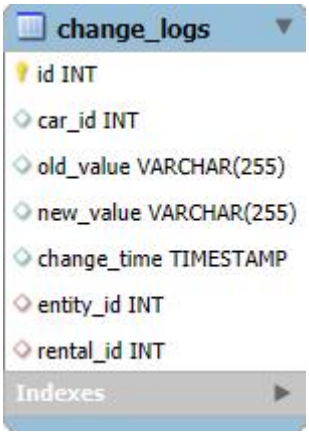

Рис.6.

- Поле id - тип INT, используется как первичный ключ.
- Поле entity_id - тип INT, используется как указание идентификатора объекта (автомобиля), у которого изменяется статус.
- Поля old_value и new_value - тип VARCHAR(255), так как они хранят текстовые значения статусов (например ‘available’).
- Поле change_time - тип TIMESTAMP, так как необходимо зафиксировать точное время изменения. Значение может задаваться автоматически при вставке записи.
- Поле tental_id - тип INT. Внешний ключ на rentals.id - связанная аренда, если изменение произошло в её контексте.
- Поле entity.id - тип INT. Внешний ключ на cars.id — идентификатор автомобиля, с которым произошло изменение.

В итоге все таблицы, имеют между собой такие связи:

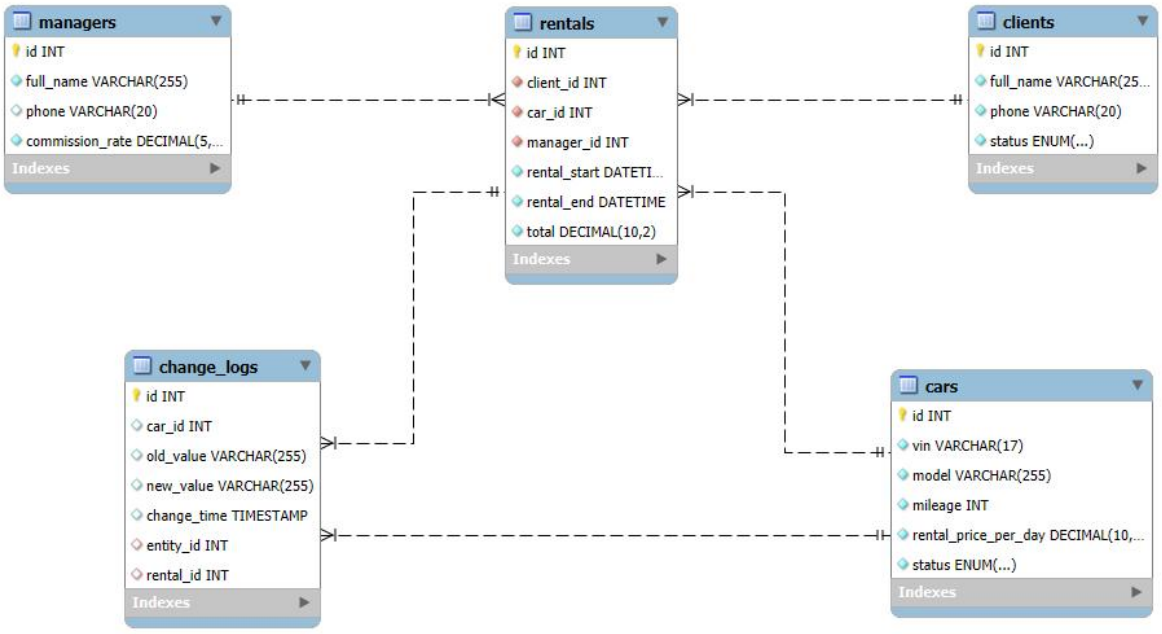

Рис.7.

Связь между таблицами

Краткое описание используемых в базе данных сервиса аренды автомобилей: Таблицы между собой связывает тип связи “один ко многим”. Такой подход обеспечивает простоту структуры, логичность взаимодействия сущностей и упрощает реализацию бизнес-логики.

Использование связей “один ко многим” позволяет чётко отразить зависимости между сущностями: один клиент может иметь несколько аренд, один менеджер - оформить множество сделок. Такой подход делает структуру базы устойчивой и легко расширяемой.

Разработка указанных выше таблиц. Воспользуемся инструментом конвертации данных MySQL Workbench и сгенерируем таблицы из ER-диаграммы

Аналогия создание таблицы cars:

```
CREATE TABLE IF NOT EXISTS `car_rental_db`.`cars` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `vin` VARCHAR(17) NOT NULL,
```

```
  `model` VARCHAR(255) NOT NULL,
  `mileage` INT NOT NULL,
  `rental_price_per_day` DECIMAL(10,2) NOT NULL,
  `status` ENUM('available', 'rented', 'maintenance') NOT NULL DEFAULT 'available',
  PRIMARY KEY (`id`),
  UNIQUE INDEX `vin` (`vin` ASC) VISIBLE
```

Пример кода для создания таблиц слишком велик что бы указывать тут (Приложение 1).

Заполнение всех таблиц тестовыми данными:

Пример заполнения таблицы managers:

```
INSERT INTO `car_rental_db`.`managers` (`full_name`, `phone`, `commission_rate`)
VALUES
('Ольга Никитина', '+7-900-123-1111', 0.02),
('Дмитрий Сидоров', '+7-900-321-2222', 0.015);
```

Код для заполнения всех таблиц (Приложение 2).

Реализовать ключевые бизнес - процессы по уровне базы данных средствами выбранной СУБД.

В рамках проекта были реализованы основные бизнес-процессы, характерные для сервиса аренды автомобилей, с использованием встроенных возможностей выбранной СУБД - MySQL. Это дозволило перенести часть логики работы системы непосредственно на уровень базы данных, повысив надёжность, целостность и автоматизацию процессов

Для этого были использованы следующие средства:

Хранимые процедуры:

1. Rent_car (Приложение 3) - оформляет аренду автомобиля:

Проверяет существование клиента, автомобиля и менеджера.

Проверяет статус машины и статус клиента (например, не находиться ли он в чёрном списке).

Выполняет расчет стоимости аренды.

Обновляет статус автомобиля и добавляет запись об аренде.

2. Return_car (Приложение 4) - оформляет возврат автомобиля:

Проверяет корректность данных, пробег и статус.

Пересчитывает стоимость аренды, если возврат произошёл позже.

Обновляет пробег автомобиля и возвращает его в статус available.

3. Blacklist (Приложение 5) - добавляет клиента в чёрный список (обновляет поле status в таблице clients на ‘blacklist’).

Пользовательская функция:

Calculate_rental_total - рассчитывает общую стоимость аренды автомобиля на основе даты начало, окончания и суточной ставки аренды.

```
CREATE DEFINER=`root`@`localhost` FUNCTION `calculate_rental_total`(
    p_price_per_day DECIMAL(10,2),
    p_start DATETIME,
    p_end DATETIME
) RETURNS decimal(10,2)
    DETERMINISTIC
BEGIN
    DECLARE v_days INT;
    DECLARE v_total DECIMAL(10,2);


    SET v_days = DATEDIFF(p_end, p_start);
    IF v_days < 1 THEN
        SET v_days = 1;
    END IF;


    SET v_total = p_price_per_day * v_days;
    RETURN v_total;
END
```

Триггер:

car_status - срабатывает после обновления статуса автомобиля:

Если статус - ‘maintenance’, то триггер автоматически добавляет запись в change_logs, фиксируя предыдущее и новое значение.

```
DELIMITER ;;

CREATE DEFINER=`root`@`localhost` TRIGGER `car_status`
AFTER UPDATE ON `cars`
FOR EACH ROW
BEGIN
   IF OLD.status != NEW.status AND NEW.status = 'maintenance' THEN
      INSERT INTO change_logs (entity_id, old_value, new_value)
      VALUES (NEW.id, OLD.status, 'maintenance');
   END IF;
END ;;

DELIMITER ;
```

Представление (view):

Rentsl_summary_view - представляет свободную информацию об арендах:

Объединяет данные из таблиц rentals, clients, cars и managers.

Упрощает отображение аренды с именами клиентов, моделей машин, именами менеджеров, суммой и сроками аренды.

```
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `rental_summary_view` AS
    SELECT
        `r`.`id` AS `rental_id`,
        `cl`.`full_name` AS `client_name`,
        `c`.`model` AS `car_model`,
        `m`.`full_name` AS `manager_name`,
        `r`.`rental_start` AS `rental_start`,
        `r`.`rental_end` AS `rental_end`,
        `r`.`total` AS `total`
    FROM
        (((`rentals` `r`
        JOIN `clients` `cl` ON ((`r`.`client_id` = `cl`.`id`)))
        JOIN `cars` `c` ON ((`r`.`car_id` = `c`.`id`)))
        JOIN `managers` `m` ON ((`r`.`manager_id` = `m`.`id`)))
```

Создание ролей:

Пользователь admin имеет полный доступ ко всем таблицам, процедурам, функциям и может управлять всей базой данных.

```
CREATE USER 'admin'@'localhost' IDENTIFIED BY 'admin_password';
GRANT ALL PRIVILEGES ON *.* TO 'admin'@'localhost';
FLUSH PRIVILEGES;
```

Пользователь manager может:

- Просматривать информацию о клиентах, машинах и аренде.
- Оформлять аренду и возврат через процедуры.
- Использовать представление или просмотра сводной информации.

Ограничения:

- Удалять данные.
- Изменять чёрный список клиентов.
- Создавать или изменять структуру таблиц и процедур.

```
CREATE USER 'manager'@'%' IDENTIFIED BY 'password';
GRANT SELECT ON `car_rental_db`.`clients` TO 'manager'@'%';
GRANT SELECT ON `car_rental_db`.`cars` TO 'manager'@'%';
GRANT SELECT ON `car_rental_db`.`rentals` TO 'manager'@'%';
GRANT EXECUTE ON PROCEDURE `car_rental_db`.`rent_car` TO 'manager'@'%';
GRANT EXECUTE ON PROCEDURE `car_rental_db`.`return_car` TO 'manager'@'%';
GRANT EXECUTE ON `car_rental_db`.`rental_summary_view` TO 'manager'@'%';
FLUSH PRIVILEGES;
```

Типовые запросы:

1. Поиск доступных автомобилей для аренды

Отображает все автомобили, которые можно арендовать прямо сейчас:

```
SELECT id, model, mileage, rental_price_per_day
FROM cars
WHERE status = 'available';
```

2. История аренд конкретного клиента

Показывает, какие машины арендовал клиент, с датами и итоговой суммой:

```
SELECT r.id AS rental_id, c.model, r.rental_start, r.rental_end, r.total
FROM rentals r
JOIN cars c ON r.car_id = c.id
WHERE r.client_id = 1;
```

3. Расчёт общей выручки за период

Считает доход от всех аренд за июнь 2025 года:

```
SELECT SUM(total) AS total_income
FROM rentals
WHERE rental_start >= '2025-06-01' AND rental_start <= '2025-06-30';
```

4. Выдача аренд, оформленных конкретным менеджером

```
SELECT m.full_name, COUNT(*) AS rentals_count, SUM(r.total) AS total_earned
FROM rentals r
JOIN managers m ON r.manager_id = m.id
GROUP BY r.manager_id;
```

5. Список автомобилей, находящихся в ремонте

Выводит машины, которые недоступны из-за техобслуживания:

```
SELECT id, model, mileage
FROM cars
WHERE status = 'maintenance';
```

Выполненные цели и задача

В ходе курсовой работы была достигнута поставленная цель - разработка базы данных для сервиса аренды автомобилей. Были выполнены следующие задачи:

- Определены бизнес-процессы предметной области.
- Сформулированы требования к базе данных.
- Выбрана СУБД MySQL.
- Простроена ER-диаграмма, отражающая структуру и связи между сущностями.
- Реализованны взаимосвязанные таблицы, процедуры, функции, триггера и представления.
- Добавлены тестовые данные и две роли пользователя.

Заключение

Разработанная база данных успешно моделирует процессы аренды автомобилей, включая оформление, возврат, учёт клиентов и автомобилей. Структура системы наглядно представлена через ER-диаграмму, что упростило реализацию и поддержку. Примером встроенных механизмов MySQL позволило реализовать ключевую логику на уровне СУБД. Система показала работоспособность и готовность к использованию.

Список литературы

1. Абрамова, Т.В., Кузнецов, С.О. База данных: проектирование, реализация и сопровождение. - М.:Форум, 2022. - 352 с.
2. Малевич, И.Е., Муромцев, Д.И Проектирование и реализация баз данных. Учебник. - М.: Академия, 2020. - 304 с.
3. Литвиненко, В.А., Киселев, С.М. Системы управления базами данных. MySQL, PostgreSQL, SQLite. - М.: БХВ-Петербург, 2022. - 416с.
4. Коряковцев, А.А Проектирование информационных систем. Учебное пособие. - М.: Инфра-М, 2021. -288 с.
5. Википедия[Электронный ресурс]/Режим доступа: https://ru.wikipedia.org/wiki/Моделирование_бизнес-процессов

Приложение 1

```
-- MySQL Script generated by MySQL Workbench
-- Mon Jun 23 00:15:45 2025
-- Model: New Model    Version: 1.0
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_I
N_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_S
UBSTITUTION';

-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------
-- -----------------------------------------------------
-- Schema car_rental_db
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema car_rental_db
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `car_rental_db` DEFAULT CHARACTER SET
utf8mb4 COLLATE utf8mb4_0900_ai_ci ;
USE `car_rental_db` ;

-- -----------------------------------------------------
-- Table `car_rental_db`.`cars`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `car_rental_db`.`cars` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `vin` VARCHAR(17) NOT NULL,
  `model` VARCHAR(255) NOT NULL,
  `mileage` INT NOT NULL,
  `rental_price_per_day` DECIMAL(10,2) NOT NULL,
  `status` ENUM('available', 'rented', 'maintenance') NOT NULL DEFAULT 'available',
  PRIMARY KEY (`id`),
  UNIQUE INDEX `vin` (`vin` ASC) VISIBLE)
ENGINE = InnoDB
AUTO_INCREMENT = 9
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;
```

```
-- -----------------------------------------------------
-- Table `car_rental_db`.`change_logs`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `car_rental_db`.`change_logs` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `entity_id` INT NULL DEFAULT NULL,
  `old_value` VARCHAR(255) NULL DEFAULT NULL,
  `new_value` VARCHAR(255) NULL DEFAULT NULL,
  `change_time` TIMESTAMP NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`))
ENGINE = InnoDB
AUTO_INCREMENT = 8
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;


-- -----------------------------------------------------
-- Table `car_rental_db`.`clients`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `car_rental_db`.`clients` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `full_name` VARCHAR(255) NOT NULL,
  `phone` VARCHAR(20) NOT NULL,
  `status` ENUM('active', 'blacklisted') NOT NULL DEFAULT 'active',
  PRIMARY KEY (`id`))
ENGINE = InnoDB
AUTO_INCREMENT = 8
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;


-- -----------------------------------------------------
-- Table `car_rental_db`.`managers`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `car_rental_db`.`managers` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `full_name` VARCHAR(255) NOT NULL,
  `phone` VARCHAR(20) NULL DEFAULT NULL,
  `commission_rate` DECIMAL(5,2) NOT NULL DEFAULT '0.01',
  PRIMARY KEY (`id`))
ENGINE = InnoDB
AUTO_INCREMENT = 7
DEFAULT CHARACTER SET = utf8mb4
```

```
COLLATE = utf8mb4_0900_ai_ci;


-- -----------------------------------------------------
-- Table `car_rental_db`.`rentals`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `car_rental_db`.`rentals` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `client_id` INT NOT NULL,
  `car_id` INT NOT NULL,
  `manager_id` INT NOT NULL,
  `rental_start` DATETIME NOT NULL,
  `rental_end` DATETIME NOT NULL,
  `total` DECIMAL(10,2) NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `client_id` (`client_id` ASC) VISIBLE,
  INDEX `car_id` (`car_id` ASC) VISIBLE,
  INDEX `manager_id` (`manager_id` ASC) VISIBLE,
  CONSTRAINT `rentals_ibfk_1`
    FOREIGN KEY (`client_id`)
    REFERENCES `car_rental_db`.`clients` (`id`),
  CONSTRAINT `rentals_ibfk_2`
    FOREIGN KEY (`car_id`)
    REFERENCES `car_rental_db`.`cars` (`id`),
  CONSTRAINT `rentals_ibfk_3`
    FOREIGN KEY (`manager_id`)
    REFERENCES `car_rental_db`.`managers` (`id`))
ENGINE = InnoDB
AUTO_INCREMENT = 7
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Приложение 2

```
INSERT INTO `car_rental_db`.`cars` (`vin`, `model`, `mileage`, `rental_price_per_day`, `status`) VALUES
('1HGCM82633A004352', 'Toyota Camry 2020', 45000, 50.00, 'available'),
('1N4AL11D75C109151', 'Nissan Altima 2019', 61000, 45.00, 'rented'),
('WBA3A5C56DF586739', 'BMW 320i 2018', 80000, 80.00, 'maintenance'),
('2C3KA63H46H239675', 'Chrysler 300 2021', 31000, 70.00, 'available'),
('3FA6P0H73HR128830', 'Ford Fusion 2022', 20000, 55.00, 'available');

INSERT INTO `car_rental_db`.`clients` (`full_name`, `phone`) VALUES
('Алексей Иванов', '+7-911-123-4567'),
('Мария Смирнова', '+7-921-765-4321'),
('Игорь Петров', '+7-931-222-3344'),
('Елена Кузнецова', '+7-912-998-7766');

INSERT INTO `car_rental_db`.`managers` (`full_name`, `phone`, `commission_rate`) VALUES
('Ольга Никитина', '+7-900-123-1111', 0.02),
('Дмитрий Сидоров', '+7-900-321-2222', 0.015);
```

Приложение 3

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `rent_car`(
    IN p_client_id INT,
    IN p_car_id INT,
    IN p_manager_id INT,
    IN p_rental_start DATETIME,
    IN p_rental_end DATETIME
)
BEGIN
    DECLARE n_client_id INT;
    DECLARE n_car_id INT;
    DECLARE n_manager_id INT;
    DECLARE status_car VARCHAR(255);
    DECLARE status_client VARCHAR(255);
    DECLARE price_per_day DECIMAL(10,2);
    DECLARE total DECIMAL(10,2);
    DECLARE days INT;
    DECLARE new_rental_id INT;

    START TRANSACTION;

    SET n_client_id = (SELECT id FROM clients WHERE id = p_client_id);
```

```
    SET n_car_id = (SELECT id FROM cars WHERE id = p_car_id);
    SET n_manager_id = (SELECT id FROM managers WHERE id = p_manager_id);

    IF (n_client_id IS NULL OR n_car_id IS NULL OR n_manager_id IS NULL) THEN
        ROLLBACK;
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Клиент, автомобиль
или менеджер не найдены';
    END IF;

    SET status_client = (SELECT status FROM clients WHERE id = p_client_id);
    IF status_client = 'blacklisted' THEN
        ROLLBACK;
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Клиент находится в
чёрном списке и не может арендовать автомобиль';
    END IF;

    SET status_car = (SELECT status FROM cars WHERE id = p_car_id);
    IF status_car != 'available' THEN
        ROLLBACK;
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Автомобиль
недоступен для аренды';
    END IF;

    SET days = DATEDIFF(p_rental_end, p_rental_start);
    IF days <= 0 THEN
        ROLLBACK;
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Некорректный период
аренды';
    END IF;

    SET price_per_day = (SELECT rental_price_per_day FROM cars WHERE id =
p_car_id);
    SET total = price_per_day * days;

    UPDATE cars
    SET status = 'rented'
    WHERE id = p_car_id;

    INSERT INTO rentals (client_id, car_id, manager_id, rental_start, rental_end, total)
    VALUES (p_client_id, p_car_id, p_manager_id, p_rental_start, p_rental_end, total);

    SET new_rental_id = LAST_INSERT_ID();

    INSERT INTO change_logs (entity_id, old_value, new_value, rental_id)
    VALUES (p_car_id, status_car, 'rented', new_rental_id);
```

```
    COMMIT;
END
```

Приложение 4

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `return_car`(
    IN p_rental_id INT,
    IN p_actual_return DATETIME,
    IN p_new_mileage INT
)
BEGIN
    DECLARE v_car_id INT;
    DECLARE v_status VARCHAR(255);
    DECLARE v_old_end DATETIME;
    DECLARE v_price_per_day DECIMAL(10,2);
    DECLARE v_total DECIMAL(10,2);
    DECLARE v_days INT;
    DECLARE v_rental_start DATETIME;
    DECLARE v_current_mileage INT;

    START TRANSACTION;


    SELECT r.car_id, c.status, r.rental_start, r.rental_end, c.rental_price_per_day,
c.mileage
    INTO v_car_id, v_status, v_rental_start, v_old_end, v_price_per_day,
v_current_mileage
    FROM rentals r
    JOIN cars c ON r.car_id = c.id
    WHERE r.id = p_rental_id;


    IF v_car_id IS NULL THEN
        ROLLBACK;
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Аренда не найдена';
    END IF;


    IF v_status = 'available' THEN
        ROLLBACK;
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Автомобиль уже
возвращён';
    END IF;
```

```
    IF p_new_mileage <= v_current_mileage THEN
        ROLLBACK;
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Неверный пробег:
меньше или равен текущему';
    END IF;


    SET v_days = DATEDIFF(p_actual_return, v_rental_start);
    IF v_days <= 0 THEN
        SET v_days = 1; -- минимум 1 день
    END IF;

    SET v_total = v_days * v_price_per_day;


    UPDATE rentals
    SET rental_end = p_actual_return,
        total = v_total
    WHERE id = p_rental_id;


    UPDATE cars
    SET status = 'available',
        mileage = p_new_mileage
    WHERE id = v_car_id;


    INSERT INTO change_logs(entity_id, old_value, new_value)
    VALUES (v_car_id, v_status, 'available');

    COMMIT;
END
```

Приложение 5

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `blacklist`(
   IN p_client_id INT
)
BEGIN
   DECLARE v_exists INT;

   SET v_exists = (SELECT COUNT(*) FROM clients WHERE id = p_client_id);

   IF v_exists = 0 THEN
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Клиент не найден';
   ELSE
      UPDATE clients
      SET status = 'blacklisted'
      WHERE id = p_client_id;
   END IF;
END
```

Отчет о проверке № 9534501

Дата выгрузки: 2025-06-25 00:37:07
Пользователь: deduhindenis32@gmail.com, ID: 9534501

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

Информация о документе

№ документа: 9534501
Имя исходного файла: Курсовая Дедюхин бд.docx
Размер файла: 0.22 МБ
Размер текста: 22281
Слов в тексте: 3279
Число предложений: 329

Информация об отчете

Дата: 2025-06-25 00:37:07 - Последний готовый отчет
Оценка оригинальности: 99%
Заимствования: 1%

99.09% 0.91%

Источники:

Доля в тексте	Ссылка

Информация о документе: