МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования

«Санкт-Петербургский политехнический университет Петра Великого»

Институт компьютерных наук и кибербезопасности
Направление подготовки:
«Математика и компьютерные науки (Искусственный интеллект и машинное обучение)»

ОТЧЁТ

по дисциплине
«Управление знаниями и технологии баз данных»

Тема курсового проекта:
Разработка базы данных сервиса аренды автомобилей

Обучающийся:
студент группы 5140201/50301
Дастанбу Матин

Преподаватель:
Попов Сергей Геннадьевич

Санкт-Петербург
2025

РЕФЕРАТ

На **31** с., **10** рисунков, **4** таблиц, **2** приложения

КЛЮЧЕВЫЕ СЛОВА: аренда автомобилей, базы данных, MySQL, ER-модель, SQL, хранимые процедуры, аналитические запросы.

В работе рассматривается проектирование и реализация реляционной базы данных для сервиса аренды автомобилей. Предметная область включает клиентов, владельцев транспортных средств, автомобили, договоры аренды и отзывы. Основная цель работы — разработка структуры базы данных, обеспечивающей целостность данных, корректную реализацию связей между сущностями и поддержку аналитических запросов.

В рамках проекта выполнен анализ предметной области, разработана концептуальная ER-модель и логическая схема базы данных. Реализация выполнена в СУБД MySQL с использованием языка SQL. Для демонстрации масштабируемости системы разработаны хранимые процедуры пошаговой генерации тестовых данных в объёме до сотен тысяч записей.

В отчёте также представлены аналитические SQL-запросы и их интерпретация с точки зрения реляционной алгебры. Разработанная база данных может быть использована как основа для реальных информационных систем сервиса аренды автомобилей.

Содержание

Введение

В современных информационных системах базы данных играют ключевую роль в автоматизации бизнес-процессов и обеспечении надёжного хранения информации. Одним из активно развивающихся направлений является цифровая аренда автомобилей, включая каршеринг и P2P-сервисы, где требуется обработка больших объёмов структурированных данных в реальном времени.

Для корректного функционирования таких сервисов необходима грамотно спроектированная база данных, обеспечивающая целостность, непротиворечивость и высокую производительность запросов. Ошибки проектирования на ранних этапах могут привести к логическим несоответствиям, дублированию данных и снижению масштабируемости системы.

Целью данной курсовой работы является разработка структуры базы данных сервиса аренды автомобилей, включая анализ предметной области, проектирование ER-модели, реализацию схемы в СУБД MySQL и программную генерацию тестовых данных.

1.1 Постановка задачи

Целью разработки является создание реляционной базы данных, предназначенной для автоматизации процессов современного сервиса аренды автомобилей. Для достижения этой цели необходимо решить следующие задачи:

- **Проанализировать предметную область**, выявив ключевые объекты (клиенты, владельцы, автомобили, договоры аренды) и основные сценарии их взаимодействия.
- **Сформировать требования к системе**, включая требования к структуре хранения данных, целостности, уникальности, безопасности и соответствию бизнес-правилам сервиса.
- **Определить набор сущностей и их атрибутов**, необходимых для представления информации о клиентах, автомобилях, владельцах и условиях аренды.
- **Описать роли участников системы**, включая клиентов-арендаторов, владельцев автомобилей и компанию-оператора, обеспечивающую процесс аренды.
- **Разработать ER-диаграмму**, отражающую основные сущности и связи между ними, а также определить типы связей и кардинальности.
- **Сформировать логическую модель данных**, на основании которой будут созданы таблицы реляционной базы данных.
- **Разработать SQL-скрипты** для создания структуры базы данных и механизмов автоматического заполнения её тестовыми данными.
- **Проверить корректность работы базы данных** с помощью тестовых запросов, демонстрирующих извлечение данных, фильтрацию, сортировку и анализ.

Результатом выполнения данной задачи должна стать рабочая база данных, способная поддерживать основные бизнес-процессы сервиса аренды автомобилей, обеспечивая удобство, надежность и масштабируемость системы.

2 . Аналитика

2.1 Описание предметной области

Современные сервисы аренды автомобилей представляют собой цифровые платформы, которые обеспечивают взаимодействие между владельцами машин, предоставляющими свои транспортные средства, и клиентами, нуждающимися в краткосрочной аренде. Основные процессы включают регистрацию пользователей, учет автомобилей, управление доступностью транспорта, оформление договоров аренды, расчёт стоимости и ведение истории операций.

Предметная область характеризуется большим количеством разнородных данных: сведения о клиентах, техническая информация об автомобилях, условия аренды, статусы договоров, история операций. Для их корректной обработки необходима система хранения данных, обеспечивающая целостность, непротиворечивость и удобство доступа.

2.2 Цели создания базы данных

Создаваемая база данных предназначена для:

- хранения полной и структурированной информации о клиентах, владельцах и автомобилях;
- автоматизации оформления и регистрации договоров аренды;
- контроля доступности автомобилей;
- предоставления инструментов для аналитики и получения статистики;
- обеспечения целостности данных и предотвращения дублирования;
- ускорения работы сервиса за счет оптимизированных запросов.

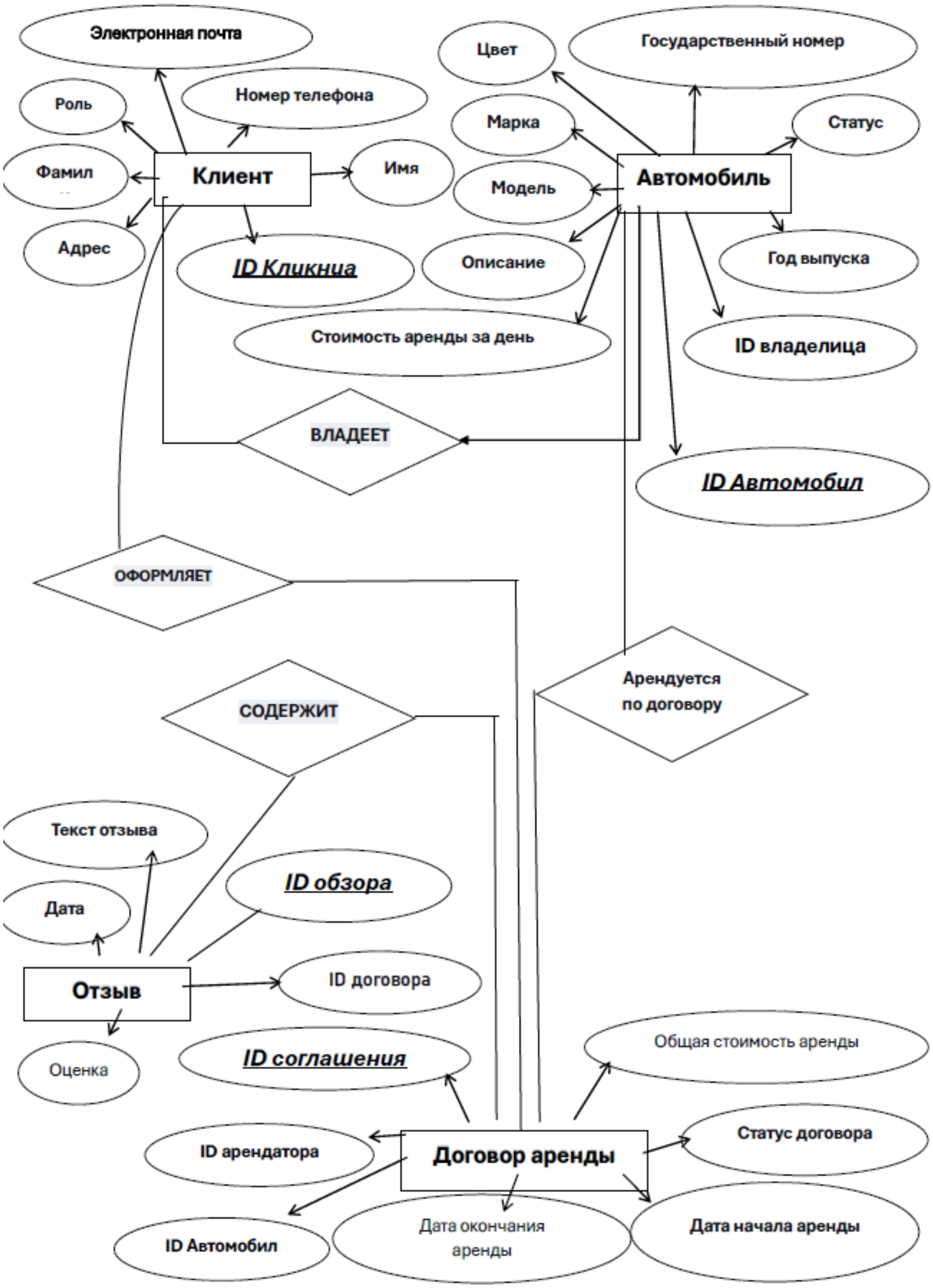
2.3 ER-модель

Данная ER-диаграмма описывает структуру базы данных для сервиса аренды автомобилей по модели "от человека к человеку" (P2P). Модель построена вокруг четырёх основных
сущностей: **CLIENT** (Клиент), **CAR** (Автомобиль), **RENTAL_AGREEMENT** (Договор аренды) и **REVIEW** (Отзыв).

Сущность **CLIENT** имеет двойную роль: атрибут "роль" (role) определяет, является ли клиент **владельцем** или **арендатором**. Владелец (который сам является CLIENT) может добавить несколько автомобилей, что создаёт связь "один-ко-многим" (**OWNS**) с сущностью **CAR**.

Сущность **RENTAL_AGREEMENT** (Договор аренды) является ядром системы. Она реализует связь "многие-ко-многим" между арендатором (CLIENT) и автомобилем (CAR). В ней записываются детали каждой аренды, такие как дата начала, дата окончания и общая стоимость.

Наконец, сущность **REVIEW** (Отзыв) связана связью "один-к-одному" с договором аренды (RENTAL_AGREEMENT). Это гарантирует, что каждый отзыв напрямую привязан к завершённой арендной сделке.


Электронная почта
Цвет
Государственный номер
Роль
Номер телефона
Марка
Статус
Фамил
Клиент
Имя
Модель
Автомобиль
Адрес
ID Кликниа
Описание
Год выпуска
Стоимость аренды за день
ID владелица
ВЛАДЕЕТ
ID Автомобил
ОФОРМЛЯЕТ
Арендуется
по договору
СОДЕРЖИТ
Текст отзыва
ID обзора
Дата
ID договора
Отзыв
Оценка
ID соглашения
Общая стоимость аренды
ID арендатора
Договор аренды
Статус договора
ID Автомобил
Дата окончания
аренды
Дата начала аренды

2.4 Упрощенная ER-модель

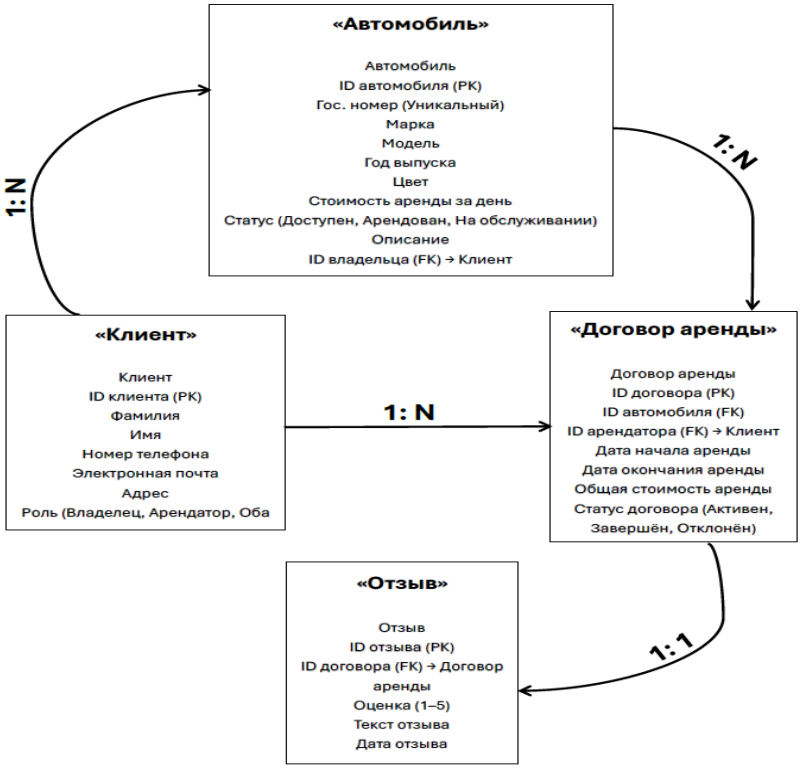
Данная диаграмма наглядно отображает структуру базы данных для сервиса аренды автомобилей. Она четко показывает четыре основные таблицы и связи между ними через внешние ключи (FK).

Основой системы является таблица **Договор аренды (Rental Agreement)**, которая выступает в роли связующего звена между **Клиентом (Client)** и **Автомобилем (Car)**. Эта структура создает связь «многие-ко-многим»: один клиент может арендовать много машин с течением времени, и одна машина может быть арендована разными клиентами

Связи между таблицами:

- Один **Клиент** может иметь несколько **Договоров аренды** (связь 1:M).
- Один **Автомобиль** может фигурировать в нескольких **Договорах аренды** (связь 1:M).
- Каждый **Договор аренды** связан с одним **Автомобилем** и одним **Арендатором** (разновидность Клиента).
- Каждый **Договор аренды** может содержать один **Отзыв** (связь 1:M, но фактически 1:1, то есть у одного договора не более одного отзыва).

Таблица **Клиент (Client)** использует поле **Роль (Role)**, чтобы различать владельцев, которые сдают машины, и арендаторов, которые их берут. Эта конструкция эффективно управляет всем жизненным циклом аренды — от размещения автомобиля до отзывов после аренды.



3 Таблицы базы данных системы проката автомобилей

3.1 CLIENT (Клиент)

Атрибут	Ключ	Тип	Описание
client_id	PK	VAR(20)	Уникальный идентификатор клиента
last_name	–	VAR(50)	Фамилия клиента
first_name	–	VAR(50)	Имя клиента
phone_number	–	VAR(15)	Номер телефона
email	–	VAR(100)	Электронная почта
address	–	TEXT	Адрес проживания
role	–	ENUM	Роль: Владелец/Арендатор/Оба

3.2 CAR (Автомобиль)

Атрибут	Ключ	Тип	Описание
car_id	PK	VAR(20)	Уникальный идентификатор автомобиля
license_plate	–	VAR(15)	Государственный номер
brand	–	VAR(50)	Марка автомобиля
model	–	VAR(50)	Модель автомобиля
year	–	INT	Год выпуска
color	–	VAR(30)	Цвет автомобиля
daily_rate	–	DEC(10,2)	Стоимость аренды за день
status	–	ENUM	Статус: Доступен/Арендован/Обслуживание
description	–	TEXT	Описание характеристик
owner_id	FK	VAR(20)	Владелец (ссылка на CLIENT)

3.3 RENTAL_AGREEMENT (Договор аренды)

Атрибут	Ключ	Тип	Описание
agreement_id	PK	VAR(20)	Уникальный идентификатор договора
car_id	FK	VAR(20)	Автомобиль (ссылка на CAR)
renter_id	FK	VAR(20)	Арендатор (ссылка на CLIENT)
start_date	–	DATE	Дата начала аренды
end_date	–	DATE	Дата окончания аренды
total_cost	–	DEC(10,2)	Общая стоимость аренды
agreement_status	–	ENUM	Статус: Активен/Завершен/Отменен

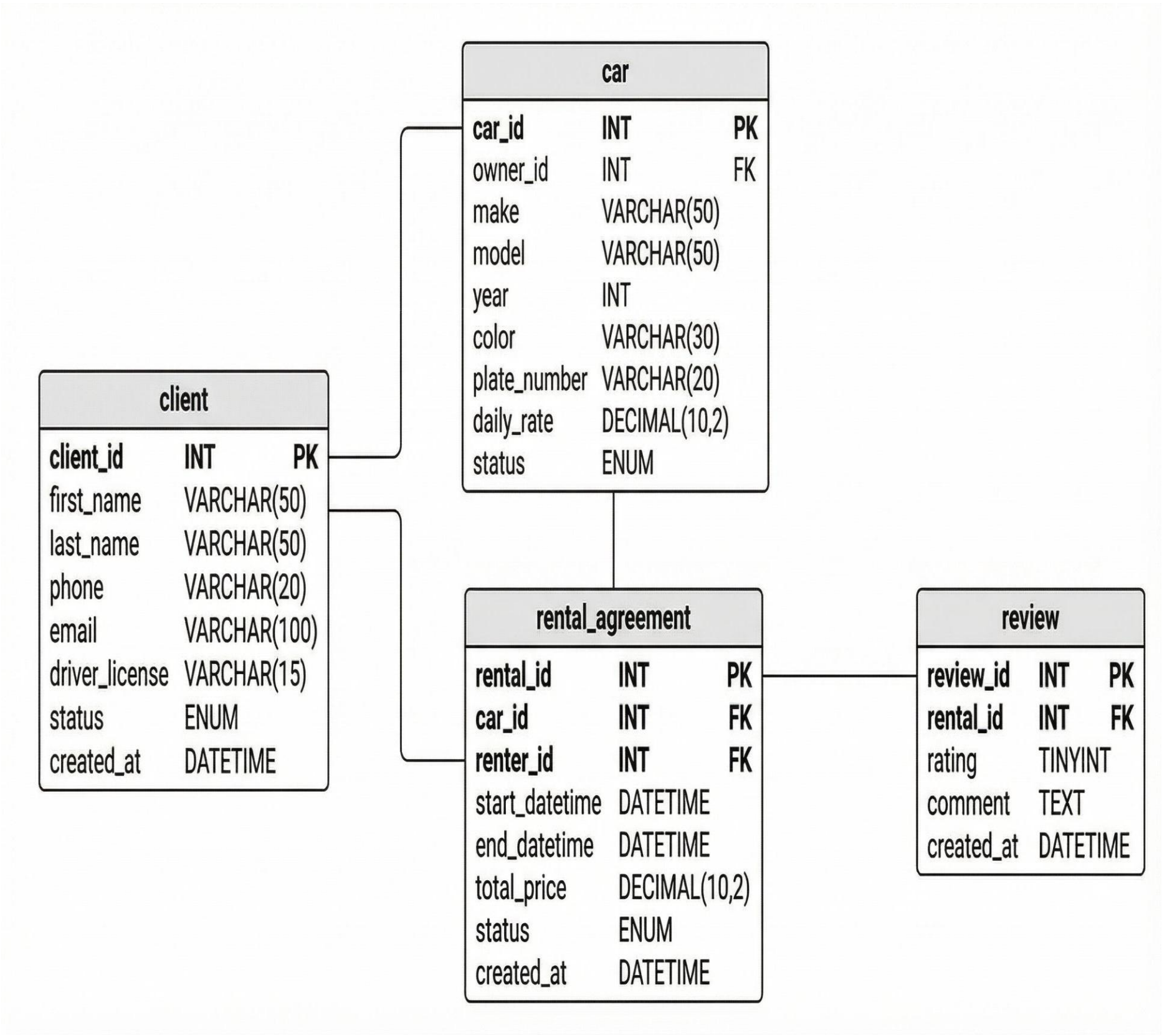
3.4 REVIEW (Отзыв)

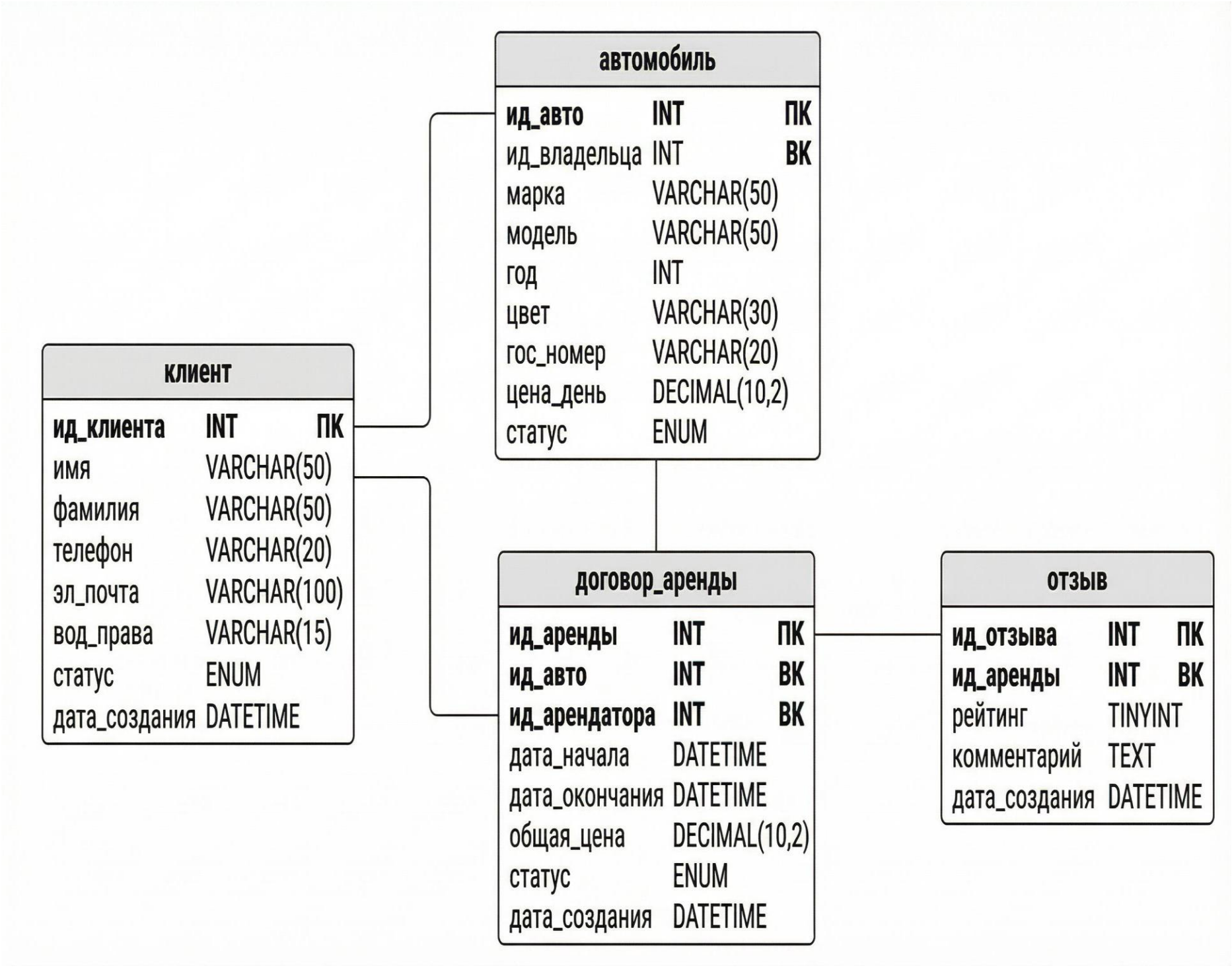
Атрибут	Ключ	Тип	Описание
review_id	PK	VAR(20)	Уникальный идентификатор отзыва
agreement_id	FK	VAR(20)	Договор (ссылка на RENTAL_AGREEMENT)
rating	–	INT	Оценка от 1 до 5 звёзд
comment_text	–	TEXT	Текст отзыва
review_date	–	DATE	Дата оставления отзыва

Обозначения типов данных:

- **VAR(n)** - Строка переменной длины (максимум n символов)
- **INT** - Целое число
- **DEC(10,2)** - Число с фиксированной точностью
- **DATE** - Дата
- **TEXT** - Текстовое поле большой длины
- **ENUM** - Перечисляемый тип данных
- **PK** - Первичный ключ
- **FK** - Внешний ключ

3.5 Схема связей объектов




автомобиль
ид_авто INT ПК
ид_владельца INT ВК
марка VARCHAR(50)
модель VARCHAR(50)
год INT
цвет VARCHAR(30)
гос_номер VARCHAR(20)
цена_день DECIMAL(10,2)
статус ENUM
клиент
ид_клиента INT ПК
имя VARCHAR(50)
фамилия VARCHAR(50)
телефон VARCHAR(20)
эл_почта VARCHAR(100)
вод_права VARCHAR(15)
статус ENUM
дата_создания DATETIME
договор_аренды
ид_аренды INT ПК
ид_авто INT ВК
ид_арендатора INT ВК
дата_начала DATETIME
дата_окончания DATETIME
общая_цена DECIMAL(10,2)
статус ENUM
дата_создания DATETIME
отзыв
ид_отзыва INT ПК
ид_аренды INT ВК
рейтинг TINYINT
комментарий TEXT
дата_создания DATETIME

4. Программирование

Раздел «Программирование» посвящён практической реализации разработанной базы данных, включая генерацию структуры БД, создание программных модулей для заполнения таблиц тестовыми данными, разработку вспомогательных процедур и выполнение SQL-запросов различной сложности. Программная часть демонстрирует работоспособность модели, корректность связей, согласованность данных и возможность использования базы данных в реальном сервисе аренды автомобилей.

4.2. Механизмы заполнения базы данных тестовыми данными

Для проверки работы системы требовалось заполнить каждую таблицу объёмом до **300 000 записей**. Поскольку выполнение больших вставок может приводить к перегрузке сервера MySQL и ошибке *"Lost connection / timeout"*, был реализован механизм поэтапного заполнения с использованием *хранимых процедур* (Stored Procedures) и *частичной загрузки (chunk-based insertion)*.

Основные принципы генерации данных:

1. **Заполнение выполняется порциями по 10 000 записей**, что предотвращает переполнение буфера сервера и снижает нагрузку.
2. **Используется вычисление текущего количества строк**, чтобы добавлять только недостающие записи.
3. **Автоматически формируются уникальные значения**, такие как:
    - номера водительских удостоверений;
    - номера автомобилей;
    - телефонные номера;
    - адреса электронной почты.
4. **Внешние ключи проверяются только после вставки данных**, а при необходимости временно отключаются (SET FOREIGN_KEY_CHECKS = 0), чтобы избежать ошибок на этапе массовой загрузки.

    Для каждого типа сущности разработана отдельная хранимая процедура:

    - fill_client_step()
    - fill_car_step()
    - fill_rental_step()
    - fill_review_step()

Эти процедуры обеспечивают автоматическую генерацию данных в соответствии с моделью и гарантируют согласованность записей между связанными таблицами.

Программные коды размещены в **Приложении В**.

4.3. Хранимые процедуры для генерации данных

Каждая процедура включает:

- объявление переменных;
- вычисление текущего количества записей;
- определение диапазона следующего блока вставки;
- генерацию значений атрибутов на основе функций MOD, CONCAT, LPAD;
- вставку данных с помощью INSERT INTO … VALUES;

- обработку уникальных и внешних ключей;
- логику перехода к следующей итерации.

Пример структуры процедуры:

```
CREATE PROCEDURE fill_client_step()
BEGIN
    DECLARE cur_count INT;
    DECLARE i INT;
    DECLARE target INT DEFAULT 300000;
    DECLARE chunk INT DEFAULT 10000;

    SELECT COUNT(*) INTO cur_count FROM CLIENT;
    SET i = cur_count + 1;

    WHILE i <= LEAST(cur_count + chunk, target) DO
        INSERT INTO CLIENT (...)
        VALUES (...);
        SET i = i + 1;
    END WHILE;
END;
```

Такой подход позволил корректно сгенерировать сотни тысяч записей без ошибок и перегрузки сервера.

4.4. SQL-запросы, реляционная алгебра и аналитический комментарий

В данном разделе представлены восемь запросов, демонстрирующих функциональные возможности разработанной базы данных системы аренды автомобилей. Каждый запрос сопровождается реляционной интерпретацией и аналитическим комментарием, что позволяет оценить не только техническую корректность SQL-решений, но и их смысловую нагрузку в контексте бизнес-модели. Во всех примерах автомобиль A определяется как автомобиль с car_id = 23.

Запрос 1. Найти клиентов, которые арендовали машину A

```
SELECT DISTINCT c.client_id,
       c.first_name,
       c.last_name,
       c.email,
       c.phone
FROM CLIENT AS c
JOIN RENTAL_AGREEMENT AS r
  ON r.renter_id = c.client_id
WHERE r.car_id = 23;
```

Реляционная алгебра

1. $R = RENTAL\_AGREEMENT \bowtie_{RENTAL\_AGREEMENT.car\_id=23}$
2. $C\_R = CLIENT \bowtie_{CLIENT.client\_id=R.renter\_id} R$
3. $\pi_{client\_id,first\_name,last\_name,email,phone}(C_R)$

Аналитический комментарий

Запрос возвращает всех клиентов, которые хотя бы один раз арендовали автомобиль с id=23. Используется внутреннее соединение, гарантирующее, что в выборку попадают только клиенты, имеющие связи в таблице аренды.

результаты

Рис 1

Запрос 2. Посчитать число аренд машины A

```
SELECT car_id,
       COUNT(*) AS rental_count
FROM RENTAL_AGREEMENT
WHERE car_id = 23
GROUP BY car_id;
```

Реляционная алгебра

1. $R = \sigma_{car\_id=23}(RENTAL\_AGREEMENT)$
2. $\gamma_{car\_id;COUNT(*)\rightarrow rental\_count}(R)$.

Аналитический комментарий

Запрос определяет количество заключённых договоров аренды по автомобилю A.
Это позволяет оценить популярность конкретной машины.

результаты

Рис 2

Запрос 3. Для первых 10 клиентов посчитать число аренд

```sql
SELECT c.client_id,
       c.first_name,
       c.last_name,
       COUNT(r.rental_id) AS rental_count
FROM (
    SELECT client_id, first_name, last_name
    FROM CLIENT
    ORDER BY client_id
    LIMIT 10
) AS c
LEFT JOIN RENTAL_AGREEMENT AS r
  ON r.renter_id = c.client_id
GROUP BY c.client_id, c.first_name, c.last_name
ORDER BY c.client_id;
```

Реляционная алгебра

1. $C_{10} = \pi_{client\_id,first\_name,last\_name}$(first 10 CLIENT)
2. C_{10}\leftouterjoin$RENTAL_AGREEMENT$
3. Группировка по client_id

Аналитический комментарий

Используется LEFT JOIN, чтобы показать клиентов без аренд (значение 0). Запрос определяет базовую активность первых десяти пользователей.

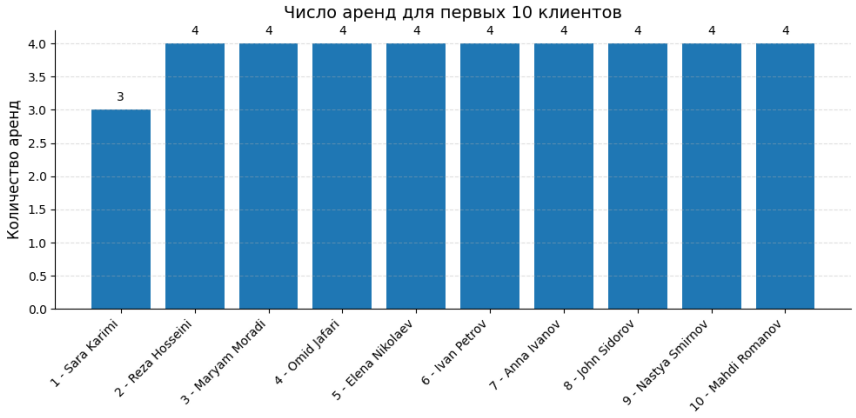


Рис 3

результаты

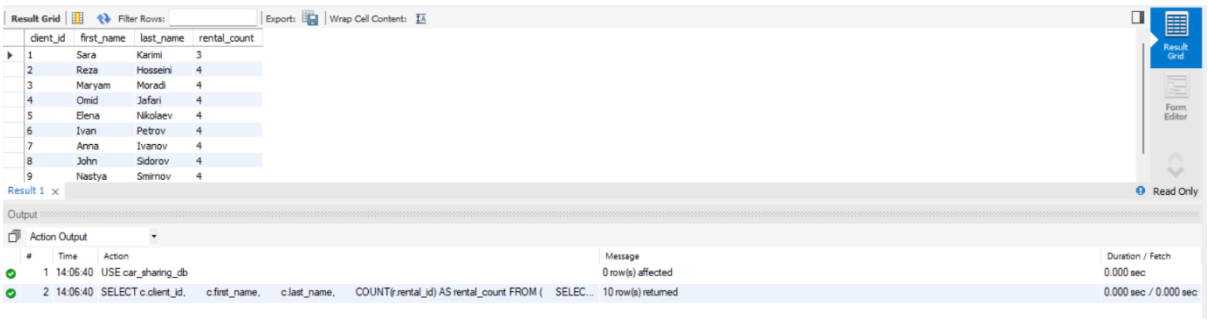


Рис 4

Запрос 4. Найти клиентов с максимальным количеством отзывов

```sql
WITH client_reviews AS (
    SELECT c.client_id,
           c.first_name,
           c.last_name,
           COUNT(v.review_id) AS review_count
    FROM CLIENT AS c
    JOIN RENTAL_AGREEMENT AS r
      ON r.renter_id = c.client_id
    JOIN REVIEW AS v
      ON v.rental_id = r.rental_id
    GROUP BY c.client_id, c.first_name, c.last_name
),
max_reviews AS (
    SELECT MAX(review_count) AS max_cnt
    FROM client_reviews
)
SELECT cr.*
FROM client_reviews AS cr
JOIN max_reviews AS mr
  ON cr.review_count = mr.max_cnt;
```

Реляционная алгебра

1. $C\_R = CLIENT \bowtie RENTAL\_AGREEMENT$
2. $C\_R\_V = C\_R \bowtie REVIEW$
3. $\gamma_{client\_id;COUNT(review\_id)}(C\_R\_V)$
4. выбор строк с максимальным значением COUNT

Аналитический комментарий

Запрос определяет наиболее активных клиентов, оставляющих максимальное число отзывов. Это помогает выявить лояльных пользователей.

результаты

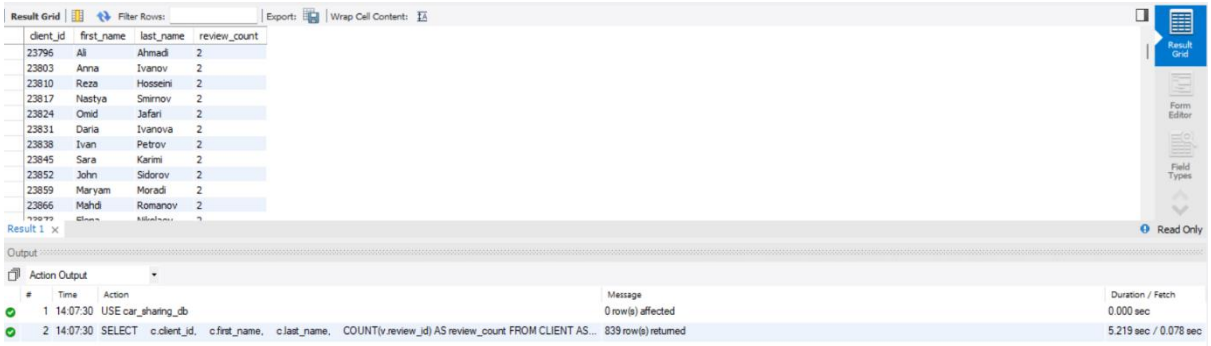


Рис 5

Запрос 5. Посчитать число клиентов с одинаковым числом аренд

```sql
SELECT rental_cnt,
       COUNT(*) AS client_count
FROM (
    SELECT c.client_id,
           COUNT(r.rental_id) AS rental_cnt
    FROM CLIENT AS c
    LEFT JOIN RENTAL_AGREEMENT AS r
      ON r.renter_id = c.client_id
    GROUP BY c.client_id
) AS t
GROUP BY rental_cnt
ORDER BY rental_cnt;
```

Реляционная алгебра

1. $C\_R = CLIENT$\leftouterjoin$RENTAL_AGREEMENT$
2. $\gamma_{client\_id;COUNT(rental\_id)}(C\_R)$
3. $\gamma_{rental\_cnt;COUNT(client\_id)}$

Аналитический комментарий

Запрос формирует распределение клиентов по количеству аренд. Это важный статистический показатель активности аудитории.

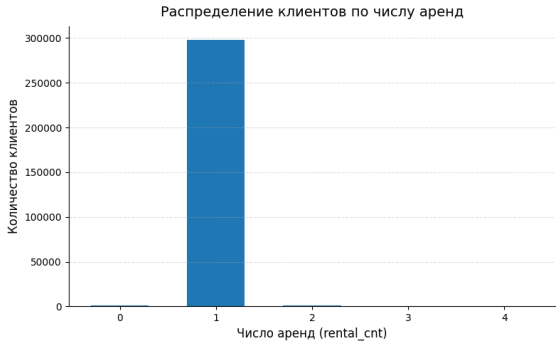


Рис 6

результаты

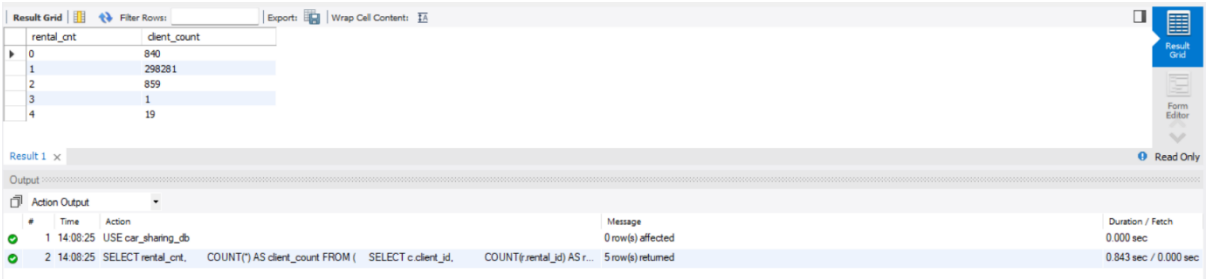


Рис 7

Запрос 6. Найти автомобили, которые арендовали чаще, чем автомобиль A (23)

```sql
SELECT c.car_id,
       c.make,
       c.model,
       COUNT(r.rental_id) AS rental_count
FROM CAR AS c
JOIN RENTAL_AGREEMENT AS r
  ON r.car_id = c.car_id
GROUP BY c.car_id, c.make, c.model
HAVING COUNT(r.rental_id) > (
    SELECT COUNT(*)
    FROM RENTAL_AGREEMENT
    WHERE car_id = 23
)
ORDER BY rental_count DESC;
```

Реляционная алгебра

1. $C\_R = CAR \bowtie RENTAL\_AGREEMENT$
2. Группировка по car_id
3. Фильтрация по значению COUNT

Аналитический комментарий

Запрос выявляет автомобили, которые используются активнее, чем автомобиль-эталон A. Полезно для анализа популярности транспортных средств.

результаты

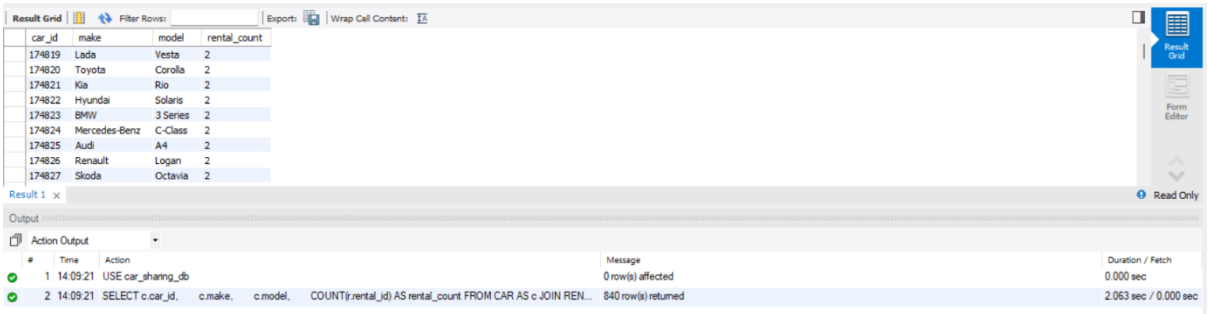


Рис 7

Запрос 7. Найти клиентов, которые ни разу не оставили отзывы

```sql
SELECT DISTINCT c.client_id,
       c.first_name,
       c.last_name
FROM CLIENT AS c
JOIN RENTAL_AGREEMENT AS r
  ON r.renter_id = c.client_id
LEFT JOIN REVIEW AS v
  ON v.rental_id = r.rental_id
WHERE v.review_id IS NULL;
```

Реляционная алгебра

1. $C\_R = CLIENT \bowtie RENTAL\_AGREEMENT$
2. $C\_R\_V = C\_R$\leftouterjoin$REVIEW$
3. $\sigma_{review\_id\ IS\ NULL}(C\_R\_V)$

Аналитический комментарий

Определяются клиенты, которые арендовали автомобили, но не оставили ни одного отзыва.
Используется LEFT JOIN для выбора клиентов без записей в таблице REVIEW.

результаты

Рис 8

Запрос 8. Для первых пяти клиентов и первых пяти машин посчитать число аренд (25 строк)

```
SELECT c.client_id,
       a.car_id,
       COUNT(r.rental_id) AS rental_count
FROM (
    SELECT client_id
    FROM CLIENT
    ORDER BY client_id
    LIMIT 5
) AS c
CROSS JOIN (
    SELECT car_id
    FROM CAR
    ORDER BY car_id
    LIMIT 5
) AS a
LEFT JOIN RENTAL_AGREEMENT AS r
  ON r.renter_id = c.client_id
 AND r.car_id    = a.car_id
GROUP BY c.client_id, a.car_id
ORDER BY c.client_id, a.car_id;
```

Реляционная алгебра

1. $C_5 =$ first 5 CLIENT
2. $A_5 =$ first 5 CAR
3. $C_5 \times A_5$(декартово произведение = 25 строк)
4. LEFT JOIN с RENTAL_AGREEMENT
5. Группировка по двум атрибутам

Аналитический комментарий

Запрос формирует таблицу 5×5, показывающую количество аренд по каждой паре «клиент–автомобиль». Позволяет оценить взаимную активность пользователей и машин.

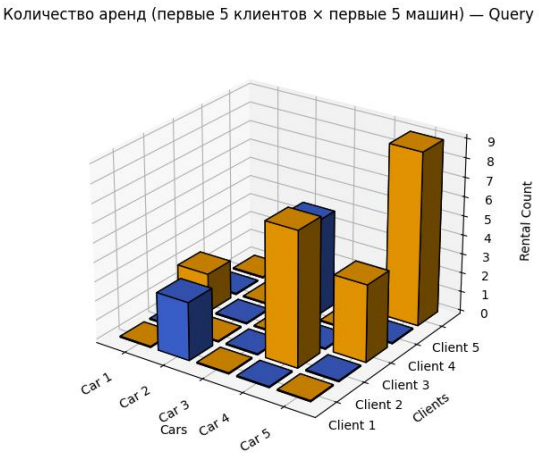


Рис 9

результаты

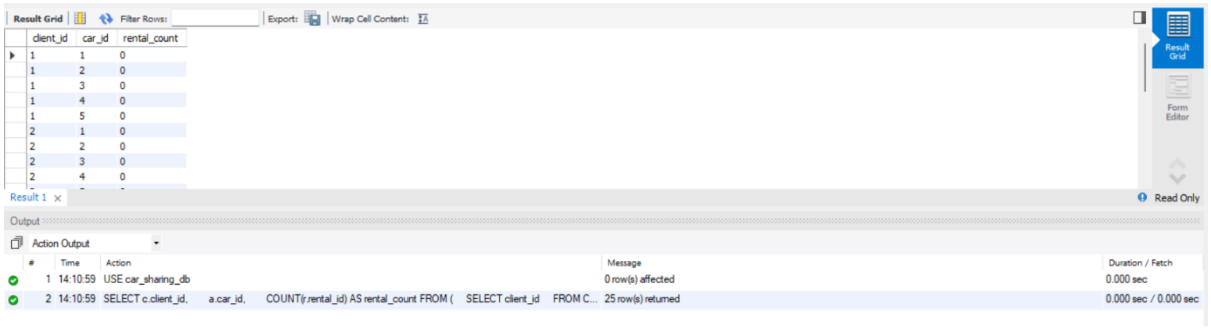


Рис 10

4.5. Реализация базы данных в MySQL

В данном разделе описывается процесс создания базы данных в СУБД MySQL на основе разработанной ER-диаграммы и логической модели данных. Реализация включала следующие этапы:

1. **Создание структуры базы данных.**
   На основе сущностей были созданы таблицы CLIENT, CAR, RENTAL_AGREEMENT и REVIEW. Каждая таблица получила первичный ключ (AUTO_INCREMENT), необходимые внешние ключи, а также ограничения NOT NULL и UNIQUE для обеспечения целостности данных.

2. **Настройка связей между таблицами.**
   - Таблица CAR содержит ссылку owner_id → CLIENT(client_id)
   - Таблица RENTAL_AGREEMENT содержит ссылки car_id → CAR(car_id) и renter_id → CLIENT(client_id)
   - Таблица REVIEW содержит ссылку rental_id → RENTAL_AGREEMENT(rental_id)

Все связи реализованы через механизм FOREIGN KEY с каскадными ограничениями, предотвращающими логические нарушения в системе.

3. **Создание индексов.**
   Были добавлены индексы по часто используемым полям (car_id, renter_id, owner_id), что повысило производительность запросов.

4. **Загрузка начальных данных.**
   Для тестирования системы были разработаны процедуры заполнения таблиц большими объемами данных — до 300 000 записей в каждой основной таблице.
   Из-за ограничения MySQL по времени выполнения операции были реализованы пошаговые процедуры (step-fill), вставляющие данные партиями по 10 000 записей.

5. **Проверка целостности и корректности данных.**
   Были выполнены тестовые запросы (см. раздел 4.4), подтверждающие корректность структуры и взаимосвязей таблиц.

Таким образом, созданная база данных является полностью работоспособной, оптимизированной и готовой для дальнейшего применения в программных системах.

4.6. Программирование механизмов заполнения базы данных

Для автоматизации генерации большого объема данных были разработаны MySQL-процедуры:

- fill_client_step() — пошаговая генерация клиентов
- fill_car_step() — пошаговая генерация автомобилей
- fill_rental_step() — пошаговая вставка договоров аренды
- fill_review_step() — генерация отзывов

Каждая процедура использует одинаковый принцип:

1. Определение количества существующих записей.
2. Вычисление диапазона значений для очередной партии вставки.
3. Генерация псевдореалистичных данных (имя, фамилия, e-mail, номер телефона, марка машины, даты и т. д.).
4. Вставка записей блоками по 10 000 строк.

Подход обеспечивает:

- отсутствие ошибок тайм-аута;
- стабильную производительность MySQL;
- отсутствие дубликатов первичных ключей;
- корректную генерацию реляционных связей.

5. Заключение

В ходе работы была спроектирована и реализована полноценная база данных для сервиса аренды автомобилей. Выполнен анализ предметной области, определены ключевые сущности и построены ER-диаграмма, логическая и физическая модели в MySQL. Реализовано автоматическое заполнение тестовыми данными для проверки масштабируемости. Результаты подтверждают корректность, надёжность и пригодность базы данных для реальных систем аренды автомобилей.

6. Список использованных источников

1. **Elmasri, R., & Navathe, S.** – *Fundamentals of Database Systems*, 7th Edition, Pearson, 2015.
   A comprehensive academic textbook covering relational databases, conceptual modeling, normalization, SQL, and DBMS architecture.

2. **Ramakrishnan, R., & Gehrke, J.** – *Database Management Systems*, 3rd Edition, McGraw-Hill, 2003.
A well-known reference for relational algebra, relational design, indexing, query optimization and transaction management.

3. **Silberschatz, A., Korth, H., & Sudarshan, S.** – *Database System Concepts*, 7th Edition, McGraw-Hill, 2019.
A widely used textbook explaining database theory, SQL, ER-modeling, and modern DBMS mechanisms.

Приложение А

Исходный код программы генерации схемы базы данных

```
CREATE DATABASE IF NOT EXISTS car_sharing_db
  DEFAULT CHARACTER SET utf8mb4
  DEFAULT COLLATE utf8mb4_unicode_ci;
USE car_sharing_db;
-- Таблица клиентов
CREATE TABLE CLIENT (
    client_id       INT AUTO_INCREMENT PRIMARY KEY,
    first_name      VARCHAR(50) NOT NULL,
    last_name       VARCHAR(50) NOT NULL,
    phone           VARCHAR(20) UNIQUE,
    email           VARCHAR(100) UNIQUE,
    driver_license  VARCHAR(20) UNIQUE,
    role            ENUM('OWNER','RENTER','BOTH') NOT NULL,
    status          ENUM('ACTIVE','BLOCKED') NOT NULL DEFAULT 'ACTIVE'
);
-- Таблица автомобилей
CREATE TABLE CAR (
    car_id       INT AUTO_INCREMENT PRIMARY KEY,
    owner_id     INT NOT NULL,
    make         VARCHAR(50) NOT NULL,
    model        VARCHAR(50) NOT NULL,
    year         INT NOT NULL,
    color        VARCHAR(30),
    plate_number VARCHAR(20) UNIQUE,
    daily_rate   INT NOT NULL,
    status       ENUM('AVAILABLE','RENTED','INACTIVE','MAINTENANCE') NOT NULL,
    FOREIGN KEY (owner_id) REFERENCES CLIENT(client_id)
);
-- Таблица договоров аренды
CREATE TABLE RENTAL_AGREEMENT (
    rental_id      INT AUTO_INCREMENT PRIMARY KEY,
    car_id         INT NOT NULL,
    renter_id      INT NOT NULL,
    start_datetime DATETIME NOT NULL,
    end_datetime   DATETIME NOT NULL,
    total_price    INT NOT NULL,
```

```
    status          ENUM('ACTIVE','COMPLETED','CANCELLED'),
    FOREIGN KEY (car_id)   REFERENCES CAR(car_id),
    FOREIGN KEY (renter_id) REFERENCES CLIENT(client_id)
);
-- Таблица отзывов
CREATE TABLE REVIEW (
    review_id   INT AUTO_INCREMENT PRIMARY KEY,
    rental_id   INT NOT NULL,
    rating      TINYINT NOT NULL,
    comment     TEXT,
    FOREIGN KEY (rental_id) REFERENCES RENTAL_AGREEMENT(rental_id)
);
```

Приложение Б

Исходный код программы заполнения базы данных тестовыми данными

```
USE car_sharing_db;
DELIMITER $$
/* ------------------------------ CLIENT (30k steps of 10k) ------------------------------ */
DROP PROCEDURE IF EXISTS fill_client_step $$
CREATE PROCEDURE fill_client_step()
BEGIN
  DECLARE i INT;
  DECLARE end_i INT;
  DECLARE cur INT;
  DECLARE chunk INT DEFAULT 10000;
  DECLARE target INT DEFAULT 300000;
  DECLARE fn VARCHAR(50);
  DECLARE ln VARCHAR(50);
  DECLARE dom VARCHAR(50);
  SELECT COUNT(*) INTO cur FROM CLIENT;
  IF cur >= target THEN LEAVE finish; END IF;
  SET i = cur + 1;
  SET end_i = LEAST(cur + chunk, target);
  WHILE i <= end_i DO
    SET fn = ELT(MOD(i,12)+1,'Ali','Sara','Reza','Maryam','Omid','Elena','Ivan','Anna','John','Nastya','Mahdi','Daria');
    SET ln =
ELT(MOD(i,12)+1,'Ahmadi','Karimi','Hosseini','Moradi','Jafari','Nikolaev','Petrov','Ivanov','Sidorov','Smirnov','Romanov','Iv
anova');
    SET dom = ELT(MOD(i,4)+1,'gmail.com','yahoo.com','mail.ru','outlook.com');
    INSERT INTO CLIENT(first_name,last_name,phone,email,driver_license,role,status)
    VALUES(
      fn,
      ln,
      CONCAT('+7-921-',LPAD(1000+i,7,'0')),
      CONCAT(LOWER(fn),'.',LOWER(ln),i,'@',dom),
      CONCAT('DL',LPAD(i,6,'0')),
      ELT(MOD(i,4)+1,'OWNER','RENTER','BOTH','RENTER'),
      IF(MOD(i,30)=0,'BLOCKED','ACTIVE')
    );
    SET i = i + 1;
  END WHILE;
```

```
  finish: BEGIN END;
END$$
/* ------------------------------ CAR (300k total) ------------------------------ */
DROP PROCEDURE IF EXISTS fill_car_step $$
CREATE PROCEDURE fill_car_step()
BEGIN
  DECLARE i INT;
  DECLARE end_i INT;
  DECLARE cur INT;
  DECLARE chunk INT DEFAULT 10000;
  DECLARE target INT DEFAULT 300000;
  DECLARE mk VARCHAR(50);
  DECLARE md VARCHAR(50);
  DECLARE clr VARCHAR(30);
  DECLARE yr INT;
  DECLARE rate INT;
  DECLARE max_cli INT;
  SELECT COUNT(*) INTO cur FROM CAR;
  SELECT MAX(client_id) INTO max_cli FROM CLIENT;
  IF cur >= target THEN LEAVE finish; END IF;
  SET i = cur + 1;
  SET end_i = LEAST(cur + chunk, target);
  WHILE i <= end_i DO
    SET mk = ELT(MOD(i,10)+1,'Toyota','Kia','Hyundai','BMW','Mercedes-
Benz','Audi','Renault','Skoda','Volkswagen','Lada');
    SET md = CASE mk
               WHEN 'Toyota' THEN 'Corolla'
               WHEN 'Kia' THEN 'Rio'
               WHEN 'Hyundai' THEN 'Solaris'
               WHEN 'BMW' THEN '3 Series'
               WHEN 'Mercedes-Benz' THEN 'C-Class'
               WHEN 'Audi' THEN 'A4'
               WHEN 'Renault' THEN 'Logan'
               WHEN 'Skoda' THEN 'Octavia'
               WHEN 'Volkswagen' THEN 'Polo'
               ELSE 'Vesta'
             END;
```

```
    SET clr = ELT(MOD(i,7)+1,'white','black','silver','blue','red','gray','green');
    SET yr = 2010 + (i MOD 15);
    SET rate = 2000 + (MOD(i,10)*150);
    INSERT INTO CAR(owner_id,make,model,year,color,plate_number,daily_rate,status)
    VALUES(
      1 + MOD(i,max_cli),
      mk,
      md,
      yr,
      clr,
      CONCAT('P',LPAD(i,6,'0')),
      rate,
      ELT(MOD(i,4)+1,'AVAILABLE','RENTED','INACTIVE','MAINTENANCE')
    );
    SET i = i + 1;
  END WHILE;
  finish: BEGIN END;
END$$
/* ------------------------------ RENTAL (300k total) ------------------------------ */
DROP PROCEDURE IF EXISTS fill_rental_step $$
CREATE PROCEDURE fill_rental_step()
BEGIN
  DECLARE i INT;
  DECLARE end_i INT;
  DECLARE cur INT;
  DECLARE chunk INT DEFAULT 10000;
  DECLARE target INT DEFAULT 300000;
  DECLARE days INT;
  DECLARE st VARCHAR(20);
  DECLARE max_car INT;
  DECLARE max_cli INT;
  DECLARE s DATETIME;
  DECLARE e DATETIME;
  SELECT COUNT(*) INTO cur FROM RENTAL_AGREEMENT;
  SELECT MAX(car_id) INTO max_car FROM CAR;
```

```
  SELECT MAX(client_id) INTO max_cli FROM CLIENT;
  IF cur >= target THEN LEAVE finish; END IF;

  SET i = cur + 1;
  SET end_i = LEAST(cur + chunk, target);
  WHILE i <= end_i DO
    SET days = 1 + MOD(i,14);
    SET s = DATE_ADD('2024-01-01 08:00:00', INTERVAL i DAY);
    SET s = DATE_ADD(s, INTERVAL MOD(i,12) HOUR);
    SET e = DATE_ADD(s, INTERVAL days DAY);
    SET st = ELT(
      CASE WHEN i<=200000 THEN 1 WHEN i<=260000 THEN 2 ELSE 3 END,
      'COMPLETED','ACTIVE','CANCELLED'
    );
    INSERT INTO RENTAL_AGREEMENT(car_id,renter_id,start_datetime,end_datetime,total_price,status)
    VALUES(
      1 + MOD(i,max_car),
      1 + MOD(i*13,max_cli),
      s,
      e,
      1500 + (days*300),
      st
    );
    SET i = i + 1;
  END WHILE;
  finish: BEGIN END;
END$$
/* ------------------------------ REVIEW (300k total) ------------------------------ */
DROP PROCEDURE IF EXISTS fill_review_step $$
CREATE PROCEDURE fill_review_step()
BEGIN
  DECLARE i INT;
  DECLARE end_i INT;
  DECLARE cur INT;
  DECLARE chunk INT DEFAULT 10000;
  DECLARE target INT DEFAULT 300000;
```

```
  DECLARE r TINYINT;
  DECLARE txt TEXT;
  SELECT COUNT(*) INTO cur FROM REVIEW;
  IF cur >= target THEN LEAVE finish; END IF;

  SET i = cur + 1;
  SET end_i = LEAST(cur + chunk, target);
  WHILE i <= end_i DO
    SET r = 1 + MOD(i + MOD(i,7), 5);
    SET txt = CASE r
      WHEN 5 THEN 'Excellent car, very clean and comfortable.'
      WHEN 4 THEN 'Good experience, recommend.'
      WHEN 3 THEN 'Average rental.'
      WHEN 2 THEN 'Some technical issues.'
      ELSE      'Bad experience.'
    END;
    INSERT INTO REVIEW(rental_id,rating,comment)
    VALUES(i,r,txt);
    SET i = i + 1;
  END WHILE;
  finish: BEGIN END;
END$$
DELIMITER ;
```