



Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Московский государственный технический университет  
имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)М

---

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА ИУ6

# **РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

## ***К КУРСОВОМУ ПРОЕКТУ***

### ***НА ТЕМУ:***

**Разработка макета аналитической системы на  
основе баз данных NoSQL (вариант № 21)**

***Прокат автомобилей***

2023 г.

**Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Московский государственный технический университет имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)**

---

УТВЕРЖДАЮ  
Заведующий кафедрой \_\_\_\_\_  
(Индекс)  
\_\_\_\_\_  
(И.О.Фамилия)  
« \_\_\_\_ » \_\_\_\_\_ 20 \_\_\_\_ г.

## **ЗАДАНИЕ на выполнение курсового проекта**

по дисциплине «Технология параллельных систем баз данных»

Тема курсового проекта

«Разработка макета аналитической системы на основе баз данных NoSQL (вариант № 24 )»

Направленность КП (учебный, исследовательский, практический, производственный, др.)  
исследовательский

Источник тематики (кафедра, предприятие, НИР): кафедра

График выполнения проекта: 25% к 3 нед., 50% к 10 нед., 75% к 13 нед., 100% к 16 нед.

### ***Задание***

Установить виртуальную машину, системы NoSQL Elasticsearch, Neo4j, Hadoop+Spark. В **Elasticsearch** создать индекс с анализатором и маппингом, проиндексировать json-документы, разработать запросы с вложенной агрегацией, представить результаты в среде Kibana. В **Neo4j** по данным из Elasticsearch заполнить графовую базу данных, разработать и реализовать запрос к этой БД. В **Spark** по данным из Elasticsearch сформировать csv-файлы (с внутренней схемой) таблиц и сохранить их в файловой системе HDFS, написать запрос и реализовать его в Spark, проанализировать процесс выполнения запроса с использованием монитора.

### ***Оформление курсового проекта:***

Расчетно-пояснительная записка на 30-50 листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.):

1. Название темы КП, задание, описание варианта.

2,3. По Elasticsearch: описание анализатора и маппинга; алгоритм программы индексации документов; тексты запросов, результаты выполнения.

4,5. По Neo4j: алгоритм программы создания и заполнения графовой БД; текст запроса, результат.

5,6,7. По Spark: алгоритм программы создания таблиц и их сохранения в HDFS; скрипт запроса к БД, результат выполнения; результат анализа работы монитора.

## РЕФЕРАТ

РПЗ 87 с., 34 рис., 1 табл., 8 источн., 3 прил.

ПРОКАТ АВТОМОБИЛЕЙБ, АРЕНДАТОР, МАШИНЫ, БАЗА ДАННЫХ, БОЛЬШИЕ ДАННЫЕ, ПАРАЛЛЕЛЬНЫЕ СИСТЕМЫ, АНАЛИТИЧЕСКАЯ СИСТЕМА, ВИЗУАЛИЗАЦИЯ ДАННЫХ, HADOOP, HDFS, KIBANA, NOSQL, SPARK.

В рамках курсового проекта была осуществлена разработка макета аналитической системы для санатория, использующей NoSQL-базы данных, такие как Elasticsearch, Neo4j, Hadoop и Spark. Реализация проекта включала следующие этапы:

- Установка виртуальной машины с операционной системой Ubuntu-20.04.3 в VirtualBox и настройка необходимых программных компонентов.
- Установка NoSQL-систем, включая Elasticsearch, Neo4j, Hadoop и Spark.
- Генерация JSON-файлов, содержащих информацию о пациентах и процедурах, для заполнения базы данных.
- Создание индекса с определением маппинга и анализатора, а также индексация JSON-документов.
- Разработка и выполнение запросов к данным, соответствующих поставленной задаче.
- Заполнение графовой базы данных и выполнение запросов в ней.
- Анализ выполнения запросов с помощью мониторинговых инструментов.

## СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ .....	5
ВВЕДЕНИЕ .....	3
1. Задание курсового проекта .....	5
2. Установка программных средств .....	8
2.1. Установка Ubuntu .....	8
2.2. Установка Elasticsearch и Kibana .....	10
2.3. Установка Neo4j .....	14
2.4. Настройка фреймворков Hadoop и Spark .....	15
3. Работа с кластером Elasticsearch .....	18
3.1. Создание JSON-документов .....	18
3.2. Индексация документов .....	19
3.3. Запросы к данным Elasticsearch .....	26
3.3.1. Первый запрос .....	26
3.3.2. Второй запрос .....	28
3.4 Neo4j .....	30
3.4.1 Создание и заполнение графовой базы данных .....	30
3.4. 2 Запрос к графовой базе данных .....	34
4. Работа со Spark .....	35
4.1. Формирование таблиц .....	35
4.2. Запрос в Spark .....	38
4.3. Мониторинг Spark .....	40
ЗАКЛЮЧЕНИЕ .....	44
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....	45
ПРИЛОЖЕНИЕ А .....	46
ПРИЛОЖЕНИЕ Б .....	67
ПРИЛОЖЕНИЕ В .....	82

## ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

Операционная система (ОС) - представляет собой программное обеспечение, которое управляет ресурсами компьютера и обеспечивает взаимодействие с пользователем.

Оперативная память (ОП) - служит для временного хранения данных и программ во время их выполнения.

База данных (БД) - представляет собой структурированное хранилище данных, доступ к которому осуществляется с помощью системы управления базами данных (СУБД).

Виртуальная машина (ВМ) - это программное или аппаратное обеспечение, которое эмулирует работу реального компьютера и позволяет запускать на нем различные операционные системы или приложения.

Directed Acyclic Graph (DAG) - это граф, в котором вершины представляют задачи или операции, а ребра обозначают зависимости между ними. Граф не содержит циклов.

ELK (Elasticsearch, Logstash, Kibana) - это стек технологий, который используется для сбора, обработки и анализа журналов и метрик данных. Он включает Elasticsearch, Logstash и Kibana.

Elasticsearch (ES) - это распределенная база данных, которая обеспечивает высокую производительность и масштабируемость при хранении и поиске данных.

Hadoop Distributed File System (HDFS) - это распределенная файловая система, которая предназначена для хранения больших объемов данных на кластерах серверов.

Распределенная база данных (DDB) - это база данных, которая распределяет данные на несколько узлов или серверов для повышения отказоустойчивости и производительности.

MapReduce - это модель распределенных вычислений, которая используется для параллельной обработки больших объемов данных. Она

разбивает задачи на более мелкие фрагменты и выполняет их параллельно на различных узлах кластера.

Стемминг - это процесс преобразования слова к его основе или корню путем удаления суффиксов и окончаний. Это позволяет упростить анализ текста, учитывая различные формы слова.

Токенизация - это процесс разбиения текста на фрагменты, которые называются токенами. Токены могут быть словами, числами, символами или другими элементами, которые являются базовыми единицами для анализа текста.

Хост-машина - это компьютер, на котором запускается другая виртуальная операционная система или виртуальная машина.

Кластер (ES) - это группа из одного или нескольких связанных узлов Elasticsearch, которые работают вместе для обеспечения отказоустойчивости, масштабируемости и обработки запросов.

Kibana - это программа для визуализации и анализа данных в Elasticsearch, которая позволяет создавать интерактивные дашборды и отчеты на основе данных из БД.

## ВВЕДЕНИЕ

В настоящее время разработка аналитических систем на основе баз данных NoSQL стала неотъемлемой частью современного мира информационных технологий. Одной из таких систем является проект "Прокат автомобилей", который направлен на создание макета аналитической системы для управления и анализа данных в сфере проката автомобилей. Для реализации этого проекта используются такие передовые технологии, как Elasticsearch, Neo4j, Hadoop и Spark.

Elasticsearch является мощной и масштабируемой распределенной базой данных, которая специально разработана для хранения, поиска и анализа больших объемов данных. Она обеспечивает высокую производительность при работе с текстовыми данными, позволяя эффективно осуществлять поиск, фильтрацию и агрегацию информации.

Neo4j, в свою очередь, представляет собой графовую базу данных, которая основана на модели направленных графов. Она идеально подходит для моделирования и анализа сложных связей и отношений между данными, что делает ее незаменимой для систем управления данными, связанными с прокатом автомобилей, такими как связи между клиентами, автомобилями, прокатными пунктами и историей аренды.

Hadoop и Spark являются платформами для обработки больших данных. Hadoop Distributed File System (HDFS) обеспечивает распределенное хранение и обработку больших объемов данных на кластерах серверов, а модель вычислений MapReduce позволяет эффективно распараллеливать вычисления над этими данными. Spark, в свою очередь, предоставляет более высокоуровневые возможности для обработки данных, включая возможности машинного обучения и анализа данных в реальном времени.

Вместе эти технологии образуют мощный стек для разработки аналитической системы "Прокат автомобилей". Они позволяют эффективно хранить, обрабатывать и анализировать большие объемы данных, а также

моделировать и исследовать сложные связи и отношения между различными аспектами проката автомобилей.

Проект включает в себя несколько этапов, начиная с установки и настройки виртуальной машины с необходимыми программными компонентами, установки и настройки систем NoSQL (Elasticsearch, Neo4j, Hadoop и Spark), генерации данных для заполнения базы данных, создания индексов и маппингов, разработки и выполнения запросов к данным, визуализации данных с помощью Kibana, а также заполнения графовой базы данных и выполнения запросов в ней. В процессе разработки аналитической системы будет осуществлен анализ выполнения запросов и мониторинг системы с использованием соответствующих инструментов.

Результатом данного проекта будет создание макета аналитической системы, способной обрабатывать и анализировать данные о прокате автомобилей с использованием передовых технологий NoSQL, что позволит эффективно управлять и извлекать ценную информацию для принятия решений в сфере проката автомобилей.



## 1. Задание курсового проекта

Для успешной реализации курсового проекта требуется выполнить несколько задач.

Вначале необходимо настроить виртуальную машину с операционной системой Ubuntu 20.04.3 в VirtualBox, учитывая определенные характеристики, такие как объем оперативной памяти и диска.

Затем следует установить необходимые системы NoSQL, включая Elasticsearch, Neo4j, Hadoop и Spark.

Для предметной области "Прокат автомобилей" требуется автоматически создать два JSON-файла с 20-30 JSON-документами каждого типа, содержащими заданные поля документов. После этого необходимо создать индекс в Elasticsearch с определенным анализатором и маппингом, а затем проиндексировать сгенерированные документы.

Для анализа данных и выполнения запросов следует разработать запросы с вложенной агрегацией и визуализировать результаты в Kibana.

Также требуется заполнить графовую базу данных Neo4j данными из Elasticsearch и разработать соответствующий запрос для выполнения операций с этой базой данных.

В рамках проекта необходимо использовать Spark для формирования таблиц в формате CSV и их сохранения в распределенной файловой системе HDFS на основе данных из Elasticsearch. Затем следует написать и выполнить запрос в Spark, а также проанализировать процесс выполнения запроса с использованием соответствующих инструментов мониторинга.

Таким образом, в курсовом проекте, связанном с предметной областью "Прокат автомобилей", требуется сгенерировать два JSON-документа с заданными полями, выполнить индексацию и обработку данных с использованием Elasticsearch, Neo4j, Hadoop и Spark, а также разработать и анализировать запросы для извлечения и анализа информации.

В 21 варианте предметной областью курсового проекта является Прокат автомобилей, для которого необходимо сгенерировать два JSON-документа с определенными полями:

Арендатор:

```
{index, doc_type, id, body: {id_арендатора, сведения_об_арендаторе*,  
id_аренды, дата_начала_аренды, продолжительность_аренды, стоимость,  
id_автомобиля}}
```

Автомобиль:

```
{index, doc_type, id, body: {сведения_об_автомобиле*,  
[диагностическая_карта_техосмотра*], [отзыв_об_автомобиле*]}}
```

Примечание. Квадратные скобки [] обозначает тег (может быть несколько значений)

Требование к анализатору: поля, отмеченные \*, разделить на слова, убрать пунктуацию с помощью токенизатора standart (русский), перевести все токены в нижний регистр, убрать токены, находящиеся в списке стоп-слов, выполнить стемминг оставшихся токенов с помощью фильтра snowball.

Помимо генерации файлов в ES требуется реализовать два запроса с вложенной агрегацией:

- разбить арендаторов по дате начала аренды с периодом 1 год, для каждой группы определить среднюю стоимость аренды по каждому автомобилю,
- определить число грузовых автомобилей в парке, используя поле «сведения\_об\_автомобиле».

Для графовой БД Neo4j необходимо сделать следующее:

1. По данным из Elasticsearch заполнить графовую базу данных Арендатор (id\_арендатора, сведения\_об\_арендаторе) - Арендатор (дата\_начала\_аренды, продолжительность\_аренды, стоимость) - Автомобиль (id\_автомобиля, сведения\_об\_автомобиле).

Примечание. В скобках приведены свойства узлов и отношения (связи), глагол – это отношение.

2. Разработать и реализовать запрос: найти автомобиль с максимальной суммарной стоимостью аренды.

Для Spark необходимо выполнить следующие задачи:

1. По данным из Elasticsearch сформировать csv-файлы (с внутренней схемой) таблиц «Арендатор», «Аренда», «Автомобиль» и сохранить их в файловой системе HDFS.

2. Написать запрос select: Определить арендатора и автомобиль с максимальной стоимостью аренды.

3. Реализовать этот запрос в Spark. Построить временную диаграмму его выполнения по результатам работы монитора.

## 2. Установка программных средств

### 2.1. Установка Ubuntu

В начале выполнения курсового проекта была установлена программа виртуализации VirtualBox. Эта среда обладает несколькими преимуществами, включая компактность, высокую производительность и бесплатное использование. В ней была развернута операционная система Linux.

С использованием VirtualBox была создана виртуальная машина с требуемыми характеристиками, согласно заданию. Для этого использовался дистрибутив Ubuntu 20.04.3 с образом ubuntu-20.04.3-desktop-amd64.iso.

Процесс установки виртуальной машины Ubuntu с операционной системой Ubuntu 20.04.3 был успешно выполнен. Работа данного процесса представлена на рисунке 1.

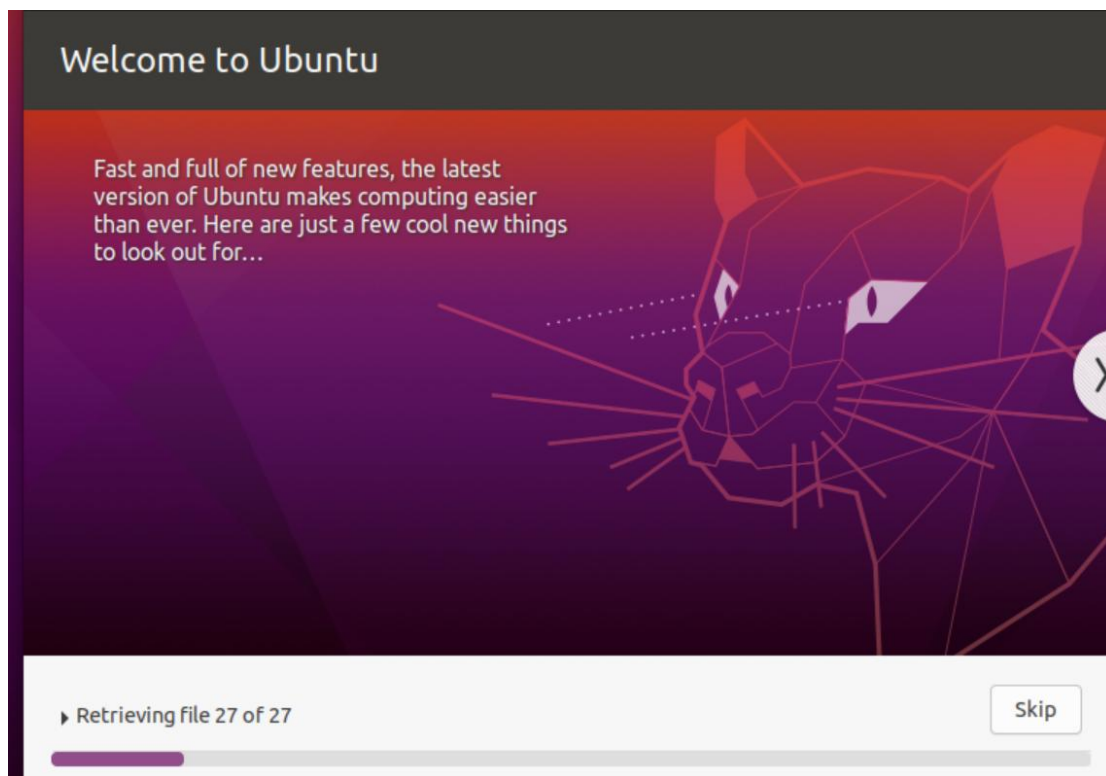


Рисунок 1 – Установка Ubuntu

После успешной установки операционной системы Ubuntu 20.30.40 в VirtualBox, в рамках проекта, была создана учетная запись с именем пользователя "hadoopuser" и паролем "hadoop". Для облегчения работы в данной виртуальной среде были настроены функции общего буфера и

Drag'n'Drop, которые позволяют копировать текст и команды между основной операционной системой и виртуальной машиной в VirtualBox. Обмен данными между ними был настроен в обоих направлениях, что позволяет использовать эти функции в обе стороны.

Визуальное представление настроек данных функций приведено на рисунке 2.

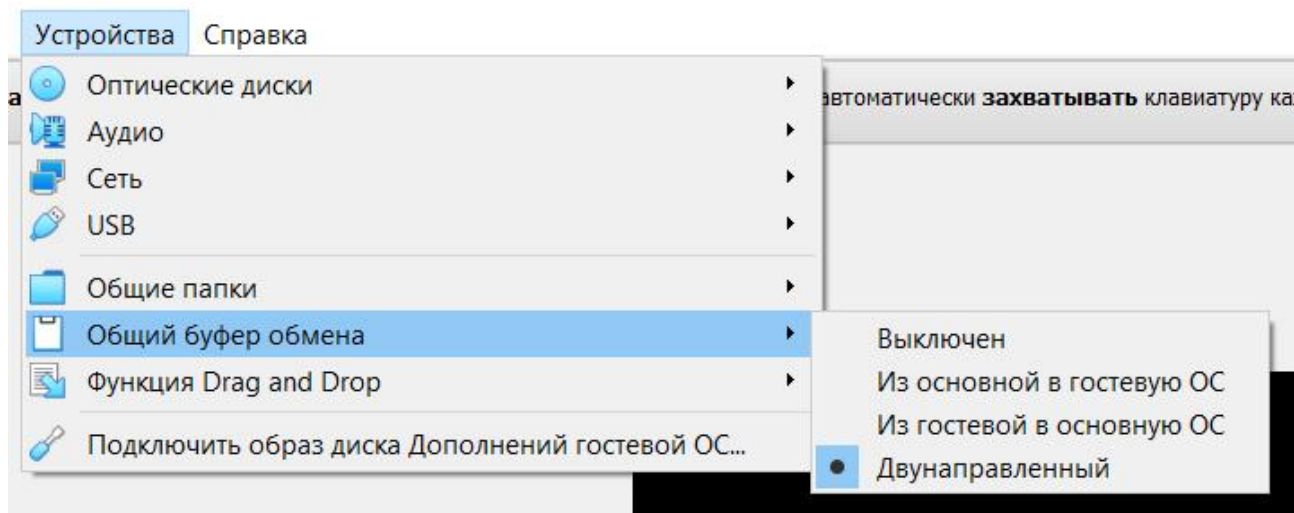


Рисунок 2 – Функции Drag'n'Drop для VirtualBox

## 2.2. Установка Elasticsearch и Kibana

Для продолжения работы была осуществлена установка Elasticsearch. Этот продукт обладает рядом значительных преимуществ, включая возможность хранения и обработки больших объемов данных, быстрый поиск и анализ информации практически в режиме реального времени. Elasticsearch базируется на мощной библиотеке Lucene, что позволяет использовать его на различных платформах. Рекомендуется установить Java версии не ниже 8, так как Lucene реализована на языке Java. На рисунке 3 показан результат выполнения команды «java -version», подтверждающий, что установлена требуемая версия Java 1.8.0.

```
sergey@sergey-VirtualBox:~$ java -version
openjdk version "1.8.0_362"
OpenJDK Runtime Environment (build 1.8.0_362-8u372-ga-us1-0ubuntu1~20.04-b09)
OpenJDK 64-Bit Server VM (build 25.362-b09, mixed mode)
```

Рисунок 3 – Версия Java

Для проверки функционирования сервера Elasticsearch требуется выполнить команду "curl -X GET localhost:9200". На рисунке 4 показан результат выполнения данной команды.

```
sergey@sergey-VirtualBox:~$ sudo systemctl start elasticsearch
sergey@sergey-VirtualBox:~$ curl -X GET "localhost:9200"
{
  "name" : "sergey-VirtualBox",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "-HwP5jVkrLCUiYFdw7najA",
  "version" : {
    "number" : "7.17.10",
    "build_flavor" : "default",
    "build_type" : "deb",
    "build_hash" : "fec6d68e3150eda0c307ab9a9d7557f5d5fd71349",
    "build_date" : "2023-04-23T05:33:18.138275597Z",
    "build_snapshot" : false,
    "lucene_version" : "8.11.1",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```

Рисунок 4 – Вывод запроса к Elasticsearch

Версия 7.17.10 Elasticsearch активна в данный момент, как подтверждает вывод команды на рисунке 4. Кроме этого, мы можем извлечь информацию о версии Elasticsearch, кластере и имени виртуальной машины. Elasticsearch можно рассматривать как хранилище NoSQL, поскольку оно способно хранить неструктурированные данные в формате JSON-файлов. Стоит также отметить, что Elasticsearch обладает и другими возможностями, такими как выполнение поисковых запросов и ряд других функциональных возможностей.

Файлы, хранящиеся в Elasticsearch, доступны для просмотра через веб-браузер, и для этой цели мы используем браузер Mozilla. В Mozilla JSON-файлы отображаются более удобно благодаря выделению и возможности скрывания отдельных разделов документа. Пример начальной страницы Elasticsearch в браузере Mozilla можно увидеть на рисунке 5.

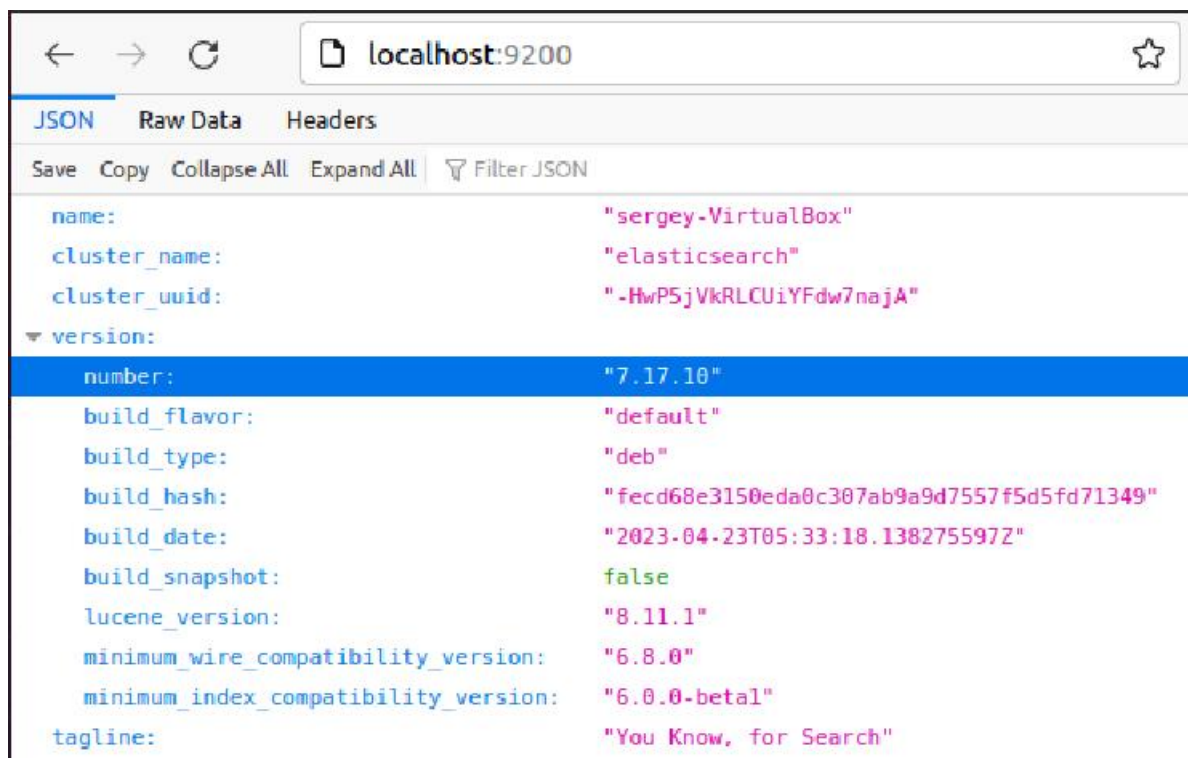


Рисунок 5 – Начальная страница Elasticsearch

Среда Kibana используется для работы с Elasticsearch. Установка Kibana выполняется через команду "sudo apt-get install kibana=x", где "x" - это версия Kibana. Важно отметить, что для каждой версии Elasticsearch требуется определенная версия Kibana. Проверить совместимость версий можно на



официальном сайте [4]. В данной работе была установлена версия 7.17.10 Kibana, так как она совместима с Elasticsearch версии 7.17.10. После запуска сервиса мы проверили его работоспособность, перейдя по адресу <http://localhost:5601/app/kibana> в браузере. Рисунок 6 демонстрирует веб-интерфейс данной среды.

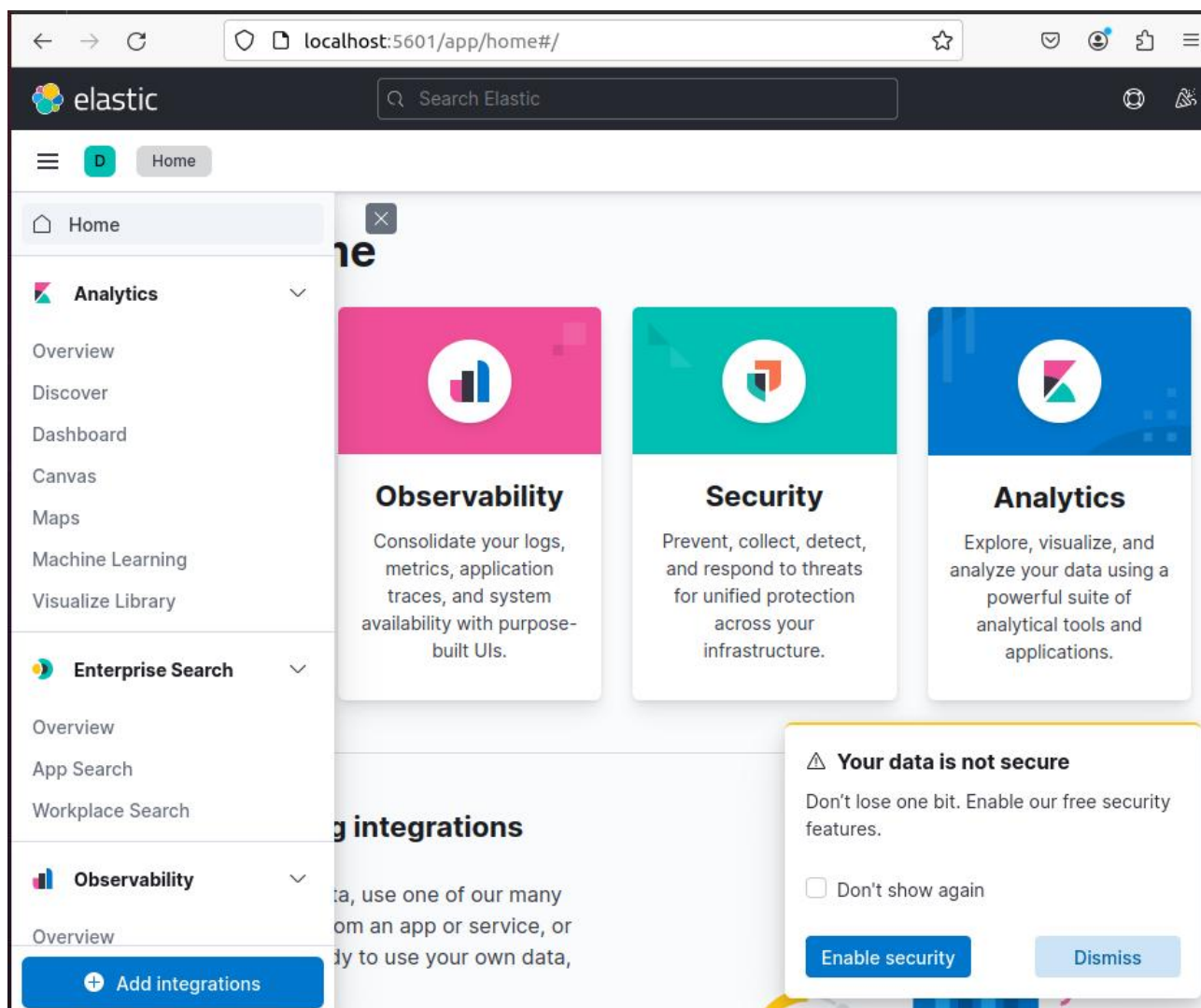


Рисунок 6 – Web-интерфейс Kibana

Kibana - это мощный инструмент для работы с данными и визуализации, который предоставляет ряд функциональных возможностей:

1. Создание Дашбордов: Kibana позволяет создавать интерактивные дашборды, на которых можно отображать различные визуализации данных, графики, таблицы и графики временных рядов.



2. Визуализации: возможность создавать разнообразные визуализации, такие как столбчатые диаграммы, круговые графики, гистограммы, карты и др., чтобы наглядно представить данные.

3. Поисковые запросы: Kibana обеспечивает мощный механизм выполнения поисковых запросов в Elasticsearch, что позволяет извлекать и анализировать данные из индексов.

4. Анализ данных: с помощью Kibana можно проводить анализ данных, выявлять тренды, аномалии и понимать структуру данных, используя инструменты, такие как фильтры и агрегации.

5. Управление индексами: возможность управлять индексами в Elasticsearch, создавать новые индексы и управлять настройками индексации данных.

6. Интерактивность: Kibana обеспечивает интерактивное взаимодействие с данными на дашбордах, что позволяет пользователям исследовать данные и принимать оперативные решения.

7. Совместимость с Elasticsearch: Kibana непосредственно интегрирован с Elasticsearch, что обеспечивает легкость доступа к данным, сохраненным в Elasticsearch.

8. Безопасность и управление доступом: Kibana предоставляет средства для управления доступом пользователей и ролей, обеспечивая безопасность данных и приложения.

9. Плагины и расширяемость: с помощью плагинов и расширений можно расширить функциональность Kibana и адаптировать его к конкретным потребностям.

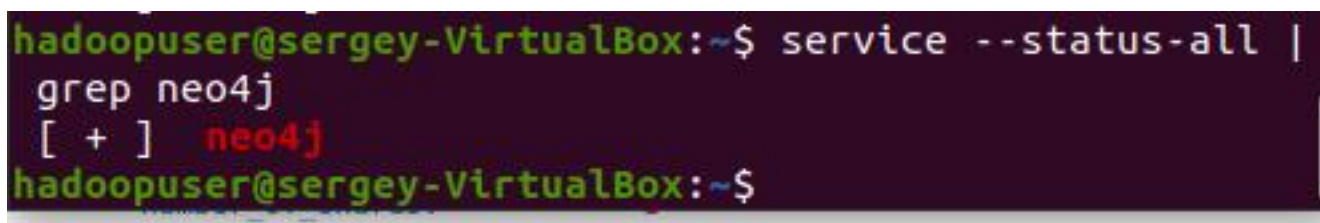
10. Интеграция с другими инструментами: Kibana может интегрироваться с другими системами и инструментами, что делает его гибким решением для анализа и визуализации данных в различных сценариях.

Таким образом, Kibana предоставляет широкий спектр возможностей для работы с данными и их визуализации, что делает его ценным инструментом для аналитики и мониторинга данных.

### 2.3. Установка Neo4j

Для установки Neo4j был выполнен последовательный набор действий. Сначала, импортирован GPG-ключ, а затем добавлен пакет Neo4j в менеджер пакетов apt. После этого, были обновлены индексы пакетов и, используя команду "sudo apt install neo4j=3.5.14", установлен сам сервис.

После удачной установки Neo4j, мы активировали его работу, выполнив команду "sudo systemctl start neo4j", и настроили автоматический запуск сервиса при каждом включении виртуальной машины с помощью команды "sudo systemctl enable neo4j". Для проверки корректности функционирования графовой СУБД воспользовались командой "service --status-all | grep neo4j", и результат подтвердил успешную установку и запуск сервиса. Результат запуска сервисов можно найти на рисунке 7.



```
hadoopuser@sergey-VirtualBox:~$ service --status-all |  
grep neo4j  
[ + ] neo4j  
hadoopuser@sergey-VirtualBox:~$
```

Рисунок 7 – Запуск Neo4j

Neo4j также предоставляет графический веб-интерфейс, который можно открыть, перейдя по адресу <http://localhost:7474>.

## 2.4. Настройка фреймворков Hadoop и Spark

Для установки Hadoop и Spark соблюдались указания и методические рекомендации из курса "Технология параллельных систем баз данных", лабораторные работы 7-8.

Сначала были загружены необходимые архивы: `hadoop-3.3.1.tar.gz` и `spark-3.2.1-bin-hadoop2.7.tgz`. После чего создана группа "hadoop", внутри которой создан пользователь "hadoopuser", предназначенный для взаимодействия с Hadoop. Далее осуществлена распаковка загруженных архивов и настройка конфигурационных файлов, обеспечивающих правильную функциональность системы [5]. Результаты выполненных команд можно увидеть на рисунке 8.

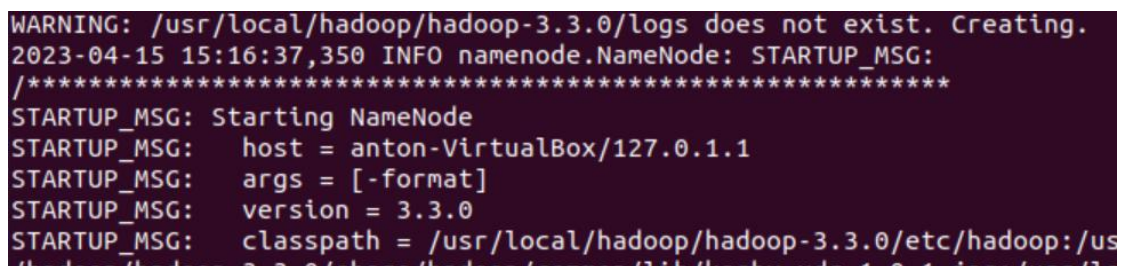


```
org/hadoop/common/hadoop-3.3.1/hadoop-3.3.1.tar.gz
--2023-04-12 04:18:43-- https://downloads.apache.org/hadoop/
common/hadoop-3.3.1/hadoop-3.3.1.tar.gz
Resolving downloads.apache.org (downloads.apache.org)... 135.
181.214.104, 88.99.95.219, 2a01:4f8:10a:201a::2, ...
Connecting to downloads.apache.org (downloads.apache.org)|135
.181.214.104|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 605187279 (577M) [application/x-gzip]
Saving to: 'hadoop-3.3.1.tar.gz'

hadoop-3.3.1.ta  4%[      ] 26,10M  1,55MB/s   eta 5m 49s
```

Рисунок 8 - Скачивание ключа для Hadoop-файлов

Далее был отформатирован Hadoop-узел при помощи команды «`hdfs namenode - format`», что представлено на рисунке 9.



```
WARNING: /usr/local/hadoop/hadoop-3.3.0/logs does not exist. Creating.
2023-04-15 15:16:37,350 INFO namenode.NameNode: STARTUP_MSG:
/*****
STARTUP_MSG: Starting NameNode
STARTUP_MSG:  host = anton-VirtualBox/127.0.1.1
STARTUP_MSG:  args = [-format]
STARTUP_MSG:  version = 3.3.0
STARTUP_MSG:  classpath = /usr/local/hadoop/hadoop-3.3.0/etc/hadoop:/us
/hadoop/hadoop-3.3.0/share/hadoop/common/lib/*:*/usr/...
```

Рисунок 9 - Форматирование Hadoop-узла

Также для корректной работы Hadoop был запущен сервер SSH и сформирован ключ RSA. Результат генерации ключа представлен на рисунке 10.

```

hadoopuser@sergey-VirtualBox:~$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hadoopuser/.ssh/id_rsa):
/home/hadoopuser/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/hadoopuser/.ssh/id_rsa
Your public key has been saved in /home/hadoopuser/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:IP1MbQeHPNYOUq/dgjoaLStwUn0F7EnwyYeg+ET6ddI hadoopuser@sergey-V
The key's randomart image is:
+---[RSA 3072]---+
| . oo . oo. |
| + . *+oo.*+ |
| o o +oEoooo+o |
| + .o++* . =.. |
| o. o S o o . |
| o . . . . |
| + o + |
| . = . |
| .o |
+---[SHA256]-----+
hadoopuser@sergey-VirtualBox:~$

```

Рисунок 10 - Формирование RSA-ключа

В результате, был создан одноузловой кластер Hadoop. Для инициализации Hadoop используются менеджеры HDFS и YARN. Для запуска HDFS используется команда «start-dfs.sh», а YARN активируется с помощью «start-yarn.sh». Однако, чтобы избежать необходимости вводить эти команды вручную, был разработан скрипт для ускоренного запуска.

Вместо двух вышеперечисленных команд необходимо ввести «./hdp.sh». Результаты выполнения данного скрипта представлены на рисунке 11.

```

hadoopuser@sergey-VirtualBox:~$ sudo chmod 777 hdp.sh
hadoopuser@sergey-VirtualBox:~$ ./hdp.sh
Starting namenodes on [localhost]
localhost: namenode is running as process 2299. Stop it first.
Starting datanodes
localhost: datanode is running as process 2437. Stop it first.
Starting secondary namenodes [sergey-VirtualBox]
sergey-VirtualBox: secondarynamenode is running as process 2619. Stop it first.
2023-09-10 19:23:35,827 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your
platform... using builtin-java classes where applicable
Starting resourcemanager
resourcemanager is running as process 2863. Stop it first.
Starting nodemanagers
localhost: nodemanager is running as process 2997. Stop it first.
hadoopuser@sergey-VirtualBox:~$

```

Рисунок 11 - Результат выполнения скрипта по запуску Hadoop



```
hadoopuser@sergey-VirtualBox:~$ jps
2437 DataNode
2997 NodeManager
3894 Jps
2619 SecondaryNameNode
2299 NameNode
2863 ResourceManager
hadoopuser@sergey-VirtualBox:~$
```

Для успешной установки Spark были внесены необходимые изменения в файл `.bashrc`. Выполнение установки Spark подтверждено и представлено на рисунке 13.

```
hadoopuser@sergey-VirtualBox:~$ spark-shell --version
23/09/10 19:16:14 WARN Utils: Your hostname, sergey-VirtualBox resolves to a l
oopback address: 127.0.1.1; using 10.0.2.15 instead (on interface enp0s3)
23/09/10 19:16:14 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to anothe
r address
Welcome to

      /_/_/_/_/_/_/_/_/_/_/_/_/_/_/_\
     / \ / \ / \ / \ / \ / \ / \ / \
    /   /   /   /   /   /   /   /   /
   /___/___/___/___/___/___/___/___/
  version 3.0.0

Using Scala version 2.12.10, OpenJDK 64-Bit Server VM, 1.8.0_362
Branch HEAD
Compiled by user ubuntu on 2020-06-06T13:05:28Z
Revision 3fdcfce3120f307147244e5eaf46d61419a723d50
Url https://gitbox.apache.org/repos/asf/spark.git
Type --help for more information.
hadoopuser@sergey-VirtualBox:~$
```

17

### 3. Работа с кластером Elasticsearch

#### 3.1.Создание JSON-документов

Для загрузки необходимых данных в кластер Elasticsearch, сначала было создано два JSON-документа: "Арендатор" и "Автомобиль". Для их создания использовались различные онлайн-сервисы. Сначала был составлен список, включающий фамилии, имена и отчества, с использованием генератора ФИО[6]. Затем, эти данные были использованы для создания двух программ, которые были обработаны онлайн-генератором JSON-документов[7].

В результате были сформированы два файла: Car.json, содержащий 30 записей, и Arendator.json, также содержащий 30 записей. Оба эти файла могут быть найдены в Приложении А.

В документе типа "Car" содержатся следующие данные:

- car\_data – сведения об автомобиле, представлено строкой;
- diagnostic\_card – перечислены неисправности, записанные в диагностическую карту техосмотра;
- car\_reviews – перечень отзывов об автомобиле.

При генерации файла "Arendator" уникальные идентификаторы "Автомобилей" были взяты из того же списка, который использовался для заполнения файла "Car".

В документе типа "Arendator" содержатся следующие данные:

- patient\_id – уникальный набор символов и цифр, позволяющий идентифицировать арендатора автомобиля;
- parentator\_data – полное ФИО арендатора ;
- arenda\_id – уникальный набор из 4 цифр, позволяющий идентифицировать номер аренды;
- date\_of\_arend – дата начала аренды;
- numb\_days\_of\_arend – количество дней, на которое арендатор взял автомобиль;
- price – стоимость аренды (в рублях);
- car\_id – id-автомобиля, которые был взят в аренду.

### 3.2. Индексация документов

Elasticsearch (ES) - это база данных, специализированная на хранении документов, где каждый документ содержит все необходимые данные и должен быть проиндексирован. Этот подход обеспечивает высокую производительность, но может создать сложности при внесении изменений в данные. В Elasticsearch, индексация документов означает добавление их в индексы.

Для управления анализом текста используется процесс маппинга, который включает в себя создание схемы данных, определяющей поля документов, подлежащие дальнейшему анализу. В данной системе также используется анализатор, который обеспечивает более эффективный поиск по текстовым данным [8].

Анализатор состоит из трех основных компонентов: символьного фильтра, токенизатора и фильтра токенов. Структура анализатора представлена на диаграмме, изображенной на рисунке 14.



Рисунок 14 – Диаграмма структуры анализатора

Приведённая структурная схема демонстрирует процесс обработки входного потока строк анализатором. Сначала символьный фильтр применяет различные операции, такие как удаление, добавление или изменение символов в строках. Затем обработанные строки передаются на вход токенизатора, который разбивает их на токены (слова). Полученные токены затем проходят через фильтр токенов и выходят на следующий этап обработки.

Для индексов "Процедура" и "Пациент" был использован анализатор, изображённый на рисунке 15. Оба JSON-документа, "Процедура" и "Пациент", были сохранены в соответствующие индексы.

```
Salon_Settings = {
  "analysis" : {
    "filter": {
      "russian_stop_words": {
        "type": "stop",
        "stopwords": "_russian_"
      },
      "filter_ru_sn": {
        "type": "snowball",
        "language": "Russian"
      }
    },
    "analyzer": {
      "analitic_for_ru": {
        "type": "custom",
        "tokenizer": "standard",
        "filter": [
          "lowercase",
          "russian_stop_words",
          "filter_ru_sn"
        ]
      }
    }
  }
}
```

Рисунок 15 – Код анализатора для индексации документов

На рисунке видно, что анализатор включает в себя токенизатор "standard", который удаляет знаки пунктуации, такие как запятые и точки. Этот анализатор предназначен для работы с русским текстом.

Сначала все токены приводятся к нижнему регистру с помощью фильтра "lowercase". Затем применяются фильтры "russian\_stop\_words" и "snowball". Фильтр "russian\_stop\_words" удаляет стоп-слова из строк, такие как союзы, предлоги и частицы. Фильтр "snowball" типа "filter\_ru\_sn" выполняет стемминг, что означает выделение основы слов и отбрасывание окончаний.



Для маппинга были указаны названия полей для документов, их типы данных, а также настроен анализатор для каждого из этих полей.

Ниже представлен пример кода маппинга для индекса "Car":

```
CarMapping = {
  "properties":{
    "car_data": {
      "type": "text",
      "analyzer":"analitic_for_ru",
      "search_analyzer":"analitic_for_ru",
      "fielddata": True
    "diagnostic_card":{
      "type": "text",
      "analyzer":"analitic_for_ru",
      "search_analyzer":"analitic_for_ru",
      "fielddata": True
    },
    "car_reviews":{
      "type": "text",
      "analyzer":"analitic_for_ru",
      "search_analyzer":"analitic_for_ru",
      "fielddata": True
    }
  }
}
```

Далее приведен код маппинга для индекса "Arendator":

```
ArendatorMapping = {
  "properties":{
    "arendator_id": {
      "type": "text",
      "fielddata": True
    }
  }
}
```

```

    },
    "arendator_data": {
        "type": "text",
        "analyzer": "analitic_for_ru",
        "search_analyzer": "analitic_for_ru",
        "fielddata": True
    },
    "arenda_id": {
        "type": "text",
        "fielddata": True
    },
    "date_of_arenda": {
        "type": "date",
        "format": "yyyy-MM-dd"
    },
    "numb_days_ofarend": {
        "type": "integer"
    },
    "price": {
        "type": "integer"
    },
    "car_id": {
        "type": "text",
        "fielddata": True
    }
}

```

В таблице 1 представлены типы данных, которые используются в маппингах индексов.

Таблица 1 – Типы данных для индексов

Тип данных	Описание типа данных
date	дата (yyyy-MM-dd )
integer	целое число
keyword	ключевые слова, используются для неизменяемых данных, таких как идентификаторы или категории
text	текстовые данные, используются для полей с текстом, подверженным анализу

Процесс индексации документов в Elasticsearch был реализован с использованием Python-скрипта. Алгоритм работы скрипта включает следующие шаги:

1. Подключение к кластеру Elasticsearch с помощью библиотеки Elasticsearch.
2. Проверка наличия индексов в кластере. Если индексы не существуют, они создаются. Если индексы уже существуют, они удаляются и создаются заново.
3. Создание анализатора и указание его настроек.
4. Закрытие индексов, обновление настроек анализатора.
5. Открытие индексов и определение полей для маппинга.
6. Добавление маппинга к индексам.
7. Открытие JSON-файлов для индексации.
8. Заполнение индексов данными из файлов.

Если в процессе выполнения возникают исключения, процесс индексации прекращается, и на экран выводится информация об ошибке.

Схема работы программы представлена на рисунке 16, а сам код программы можно найти в приложении Б.

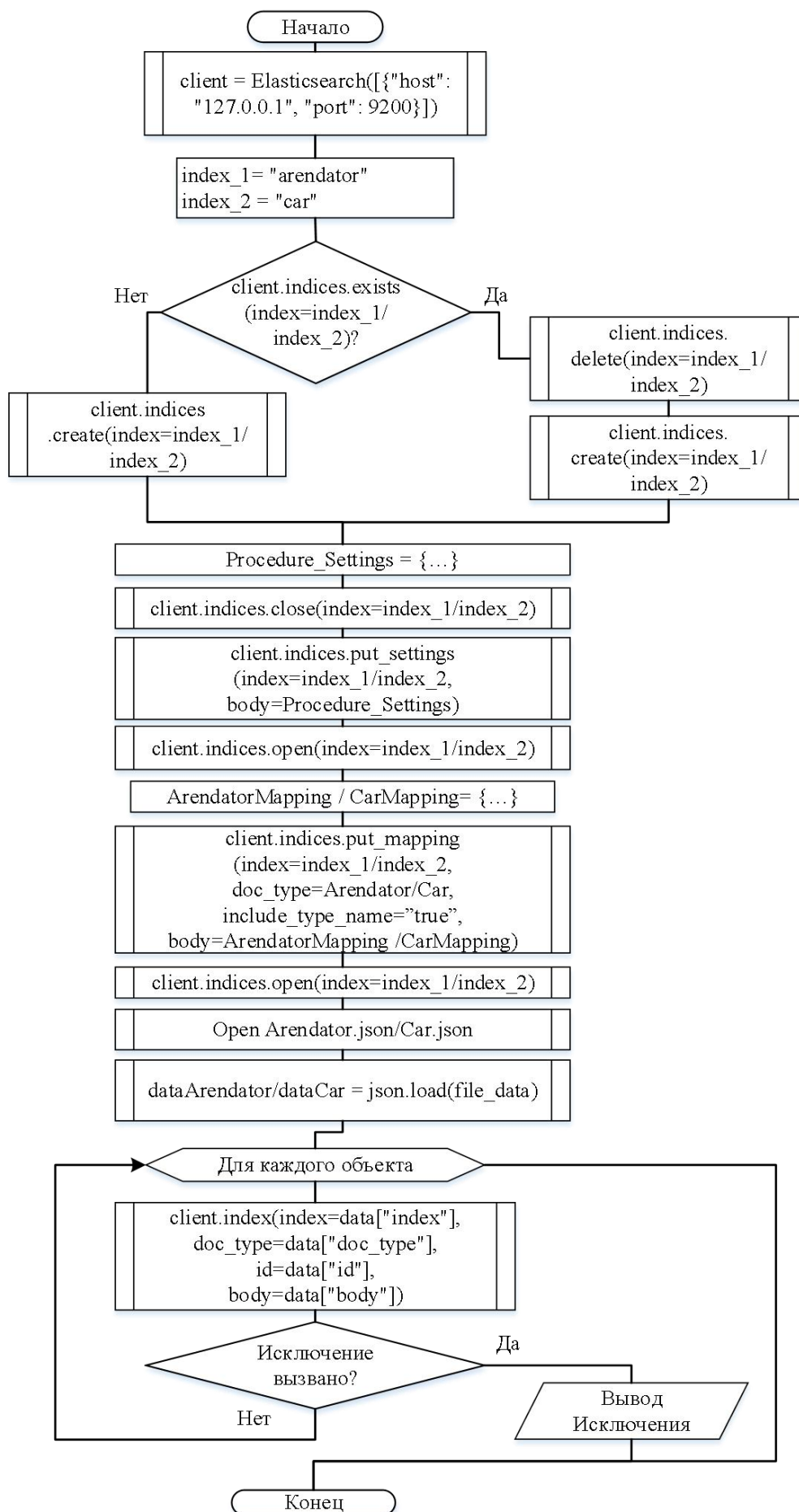


Рисунок 16 – Схема алгоритма маппинга и индексации

Результат выполнения программы представлен на рисунке 17.

```
Warning: DeprecationWarning: The 'b
index' API and will be removed in a futu
parameter. See https://github.com/elast
re information
client.index(index=data["index"],
Car_indexed
hadoopuser@sergey-VirtualBox:~$
```

Рисунок 17 – Итог выполнения программы маппинга и индексации

Для просмотра информации о индексах, их маппингах и анализаторах, можно перейти по следующим ссылкам:

1. Для индекса "Арендатор": <http://localhost:9200/arendator>
2. Для индекса "Машина": <http://localhost:9200/car>

Результаты перехода по этим ссылкам можно увидеть на рисунке 18.

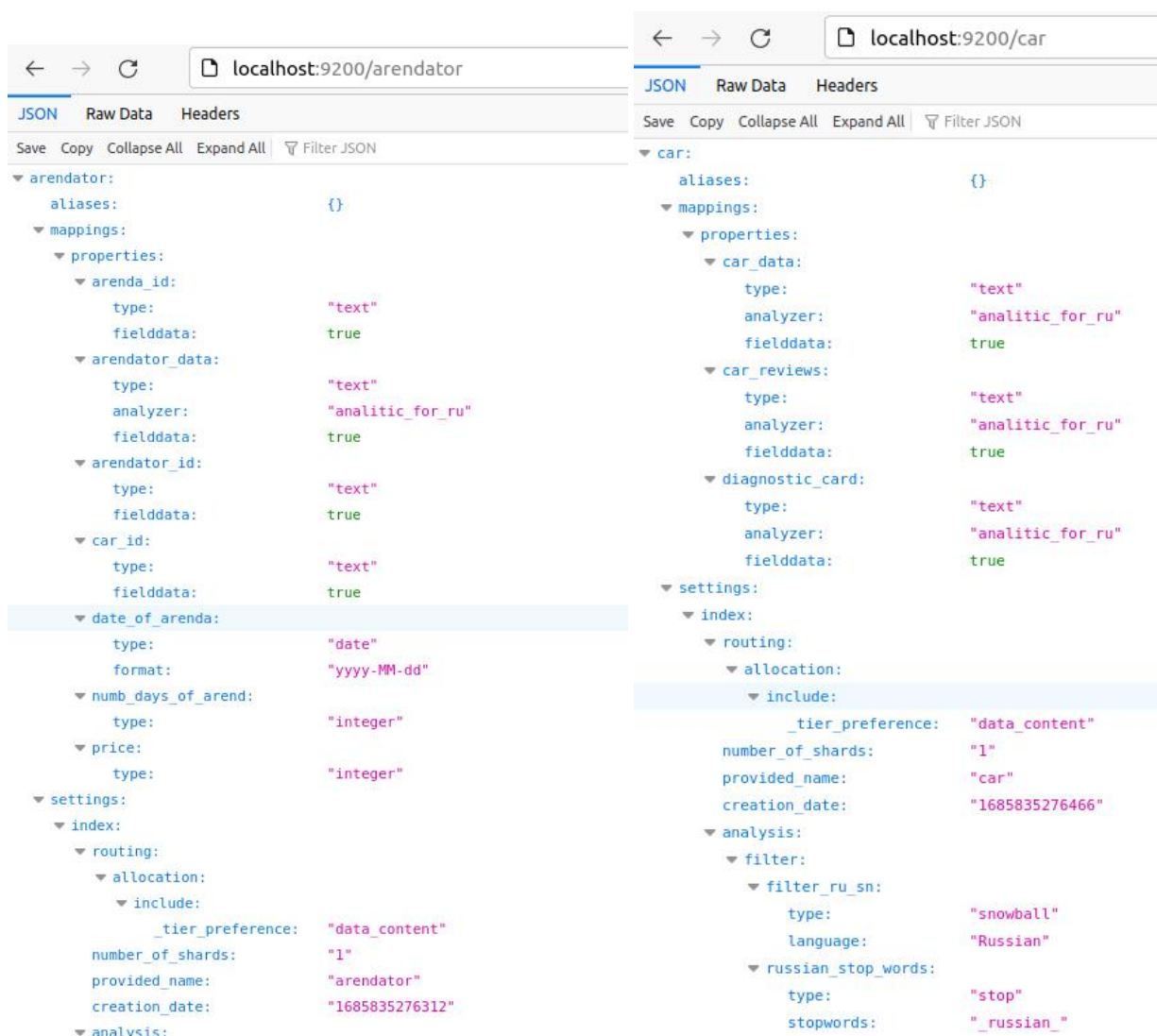


Рисунок 18 – Индексы car и arendator с маппингом и анализатором

### 3.3. Запросы к данным Elasticsearch

#### 3.3.1. Первый запрос

Согласно предоставленному заданию, необходимо создать запрос.

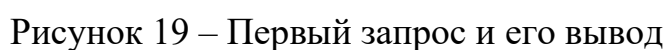
Запрос: разбить арендаторов по дате начала аренды с периодом 1 год, для каждой группы определить среднюю стоимость аренды по каждому автомобилю.

Для выполнения запроса была использована следующая реализация:

GET arendator/\_search

```
{
  "size": 0,
  "aggregations": {
    "year_period": {
      "date_histogram": {
        "field": "date_of_arend",
        "calendar_interval": "year",
        "format": "yyyy-MM-dd"
      },
    },
    "aggregations": {
      "Arendator": {
        "terms": {
          "field": "car_id",
          "order": {
            "_key": "asc"
          }
        },
      },
    },
    "aggregations": {
      "mean_arend_price": {
        "avg": {
          "field": "price"
        }
      }
    }
  }
}
```

На следующем уровне группируются процедуры для каждого года на основе их идентификатора. Завершая цепочку агрегации, на последнем уровне определяется стоимость выполненных процедур для каждого специалиста с использованием гистограммы "value\_count" по полю "patient\_id". Результат запроса, выполненного в среде Kibana, представлен на рисунке 19. Полный текст запроса можно найти в приложении А.



### 3.3.2. Второй запрос

Необходимо выполнить следующее задание: определить число грузовых автомобилей в парке, используя поле «сведения\_об\_автомобиле».

Для поиска грузовых автомобилей в описании машин была использована агрегация на основе "match\_phrase", где ключевым словом стало "грузовик".

Результат запроса содержит id автомобиля и его полное описание. Ниже приведён реализованный запрос.

```
GET car/_search
{
  "query": {
    "match_phrase": {
      "car_data": "грузовик"
    }
  },
  "_source": ["car_data"],
  "aggs": {
    "count": {
      "value_count": {
        "field": "_id"
      }
    }
  }
}
```

На рисунке под номером 20 представлен результат выполненного запроса в интерфейсе Kibana. Для полного текста запроса можно обратиться к приложению А.



```
History Settings Help 200 - OK 128 ms

1 GET car/_search
2 {
3   "query": {
4     "match_phrase": {
5       "car_data": "грузовик"
6     }
7   },
8   "_source": ["car_data"],
9   "aggs": {
10    "count": {
11      "value_count": {
12        "field": "_id"
13      }
14    }
15  }
16 }
17
18

89 {
90   },
91   {
92     "_index": "car",
93     "_type": "Car",
94     "_id": "CR22",
95     "_score": 0.7457469,
96     "_source": {
97       "car_data": "Грузовик: GMC Canyon, 2017г"
98     }
99   },
100  },
101  {
102    "_index": "car",
103    "_type": "Car",
104    "_id": "CR24",
105    "_score": 0.7457469,
106    "_source": {
107      "car_data": "Грузовик: Toyota Tundra, 2014г"
108    }
109  }
110 },
111 "aggregations": {
112   "count": {
113     "value": 15
114   }
115 }
```

Рисунок 20 – Второй запрос и его вывод

### 3.4 Neo4j

#### 3.4.1 Создание и заполнение графовой базы данных

Для заполнения графовой базы данных Neo4j были получены данные из Elasticsearch, а затем созданы узлы. В итоговой графовой базе данных были созданы следующие сущности:

- Узел: «Car\_Node» (Автомобиль): «Car\_id» (id\_автомобиля), «Car\_data» (сведения\_об\_автомобиле), «Diagnostic\_card» (диагностическая\_карта\_техосмотра), «Car\_reviews» (отзывы\_об\_автомобиле);
- Узел: «Arendator\_Node» (Арендатор): «Arendator\_id» (id\_арендатора), «Arendator\_data» (сведения\_об\_арендаторе).
- Связь: «Lent\_car» (Арендовал): «Arenda\_id» (id\_аренды), «Date\_of\_arend» (дата\_начала\_аренды), «Numb\_of\_days» (продолжительность\_аренды), «Price» (стоимость).

Для заполнения базы данных Neo4j была написана программа на языке Python, которая имеет следующую структуру:

- 1) Устанавливается подключение к кластеру Elasticsearch и клиенту Neo4j.
- 2) Существующая база данных Neo4j удаляется.
- 3) Данные импортируются из кластера Elasticsearch.
- 4) В основном цикле создается узел "Арендатор".
- 5) Во вложенном цикле проверяется наличие автомобиля в базе данных, и создаётся новый автомобиль при его отсутствии.
- 6) Создается новая связь, если идентификатор автомобиля во внутреннем цикле совпадает с одним из идентификаторов car\_id во внешнем цикле.

Схема алгоритма программы для заполнения базы данных Neo4j представлена на рисунке 21. Код программы можно найти в приложении Б.

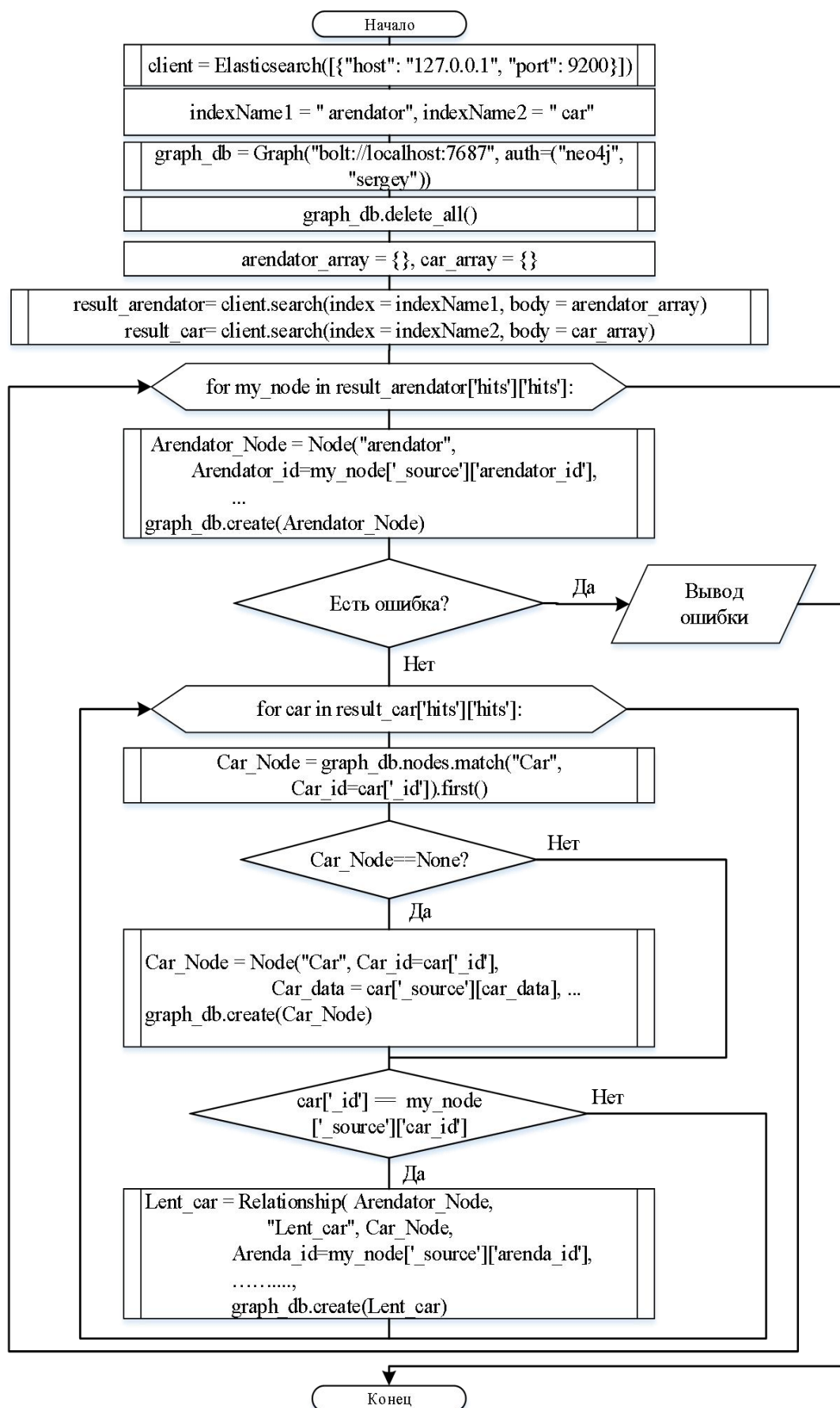
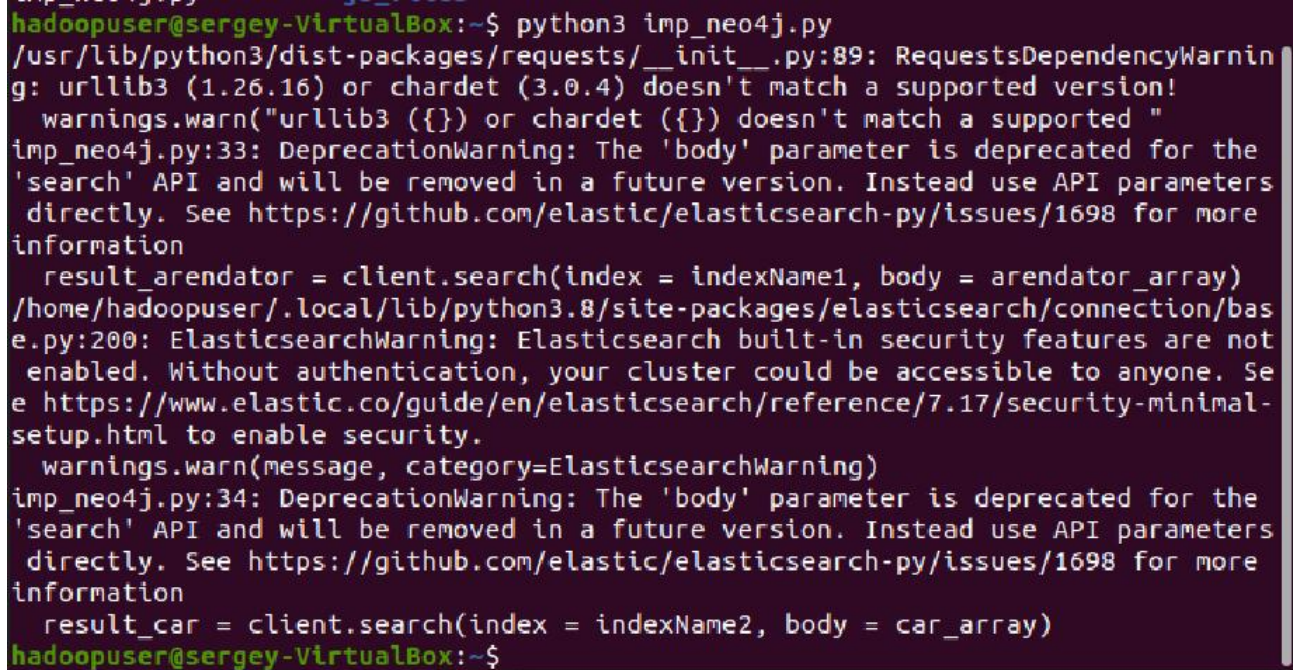


Рисунок 21 – Схема алгоритма формирования графовой базы данных Neo4j.

Результат выполнения программы представлен на рисунке 22.



```
hadoopuser@sergey-VirtualBox:~$ python3 imp_neo4j.py
/usr/lib/python3/dist-packages/requests/__init__.py:89: RequestsDependencyWarning: urllib3 (1.26.16) or chardet (3.0.4) doesn't match a supported version!
  warnings.warn("urllib3 ({}), or chardet ({}), doesn't match a supported version".format(urllib3.__version__, chardet.__version__), RequestsDependencyWarning)
imp_neo4j.py:33: DeprecationWarning: The 'body' parameter is deprecated for the 'search' API and will be removed in a future version. Instead use API parameters directly. See https://github.com/elastic/elasticsearch-py/issues/1698 for more information
  result_arendator = client.search(index = indexName1, body = arendator_array)
/home/hadoopuser/.local/lib/python3.8/site-packages/elasticsearch/connection/base.py:200: ElasticsearchWarning: Elasticsearch built-in security features are not enabled. Without authentication, your cluster could be accessible to anyone. See https://www.elastic.co/guide/en/elasticsearch/reference/7.17/security-minimal-setup.html to enable security.
  warnings.warn(message, category=ElasticsearchWarning)
imp_neo4j.py:34: DeprecationWarning: The 'body' parameter is deprecated for the 'search' API and will be removed in a future version. Instead use API parameters directly. See https://github.com/elastic/elasticsearch-py/issues/1698 for more information
  result_car = client.search(index = indexName2, body = car_array)
hadoopuser@sergey-VirtualBox:~$
```

Рисунок 22 – Результат выполнения скрипта по формированию графовой базы данных Neo4j.

Для просмотра полученной базы данных, нужно сделать переход по адресу <http://localhost:7474/browser>. Чтобы найти связь "Арендатор-Арендовал-Автомобиль", которую необходимо было создать в соответствии с заданием, был выполнен запрос на языке Cypher:

```
MATCH p=()-[r:Lent_car]->() RETURN p LIMIT 25.
```

Результат запроса представлен на рисунке 23. На данном рисунке узлы типа "Автомобиль" имеют оранжевый цвет, а узлы типа "Арендатор" - фиолетовый.

Стоит отметить, что в правой части рисунка выведены данные об узле, который был выделен в основной части. В данном случае был выделен узел Автомобиль с номером `id = 8`.

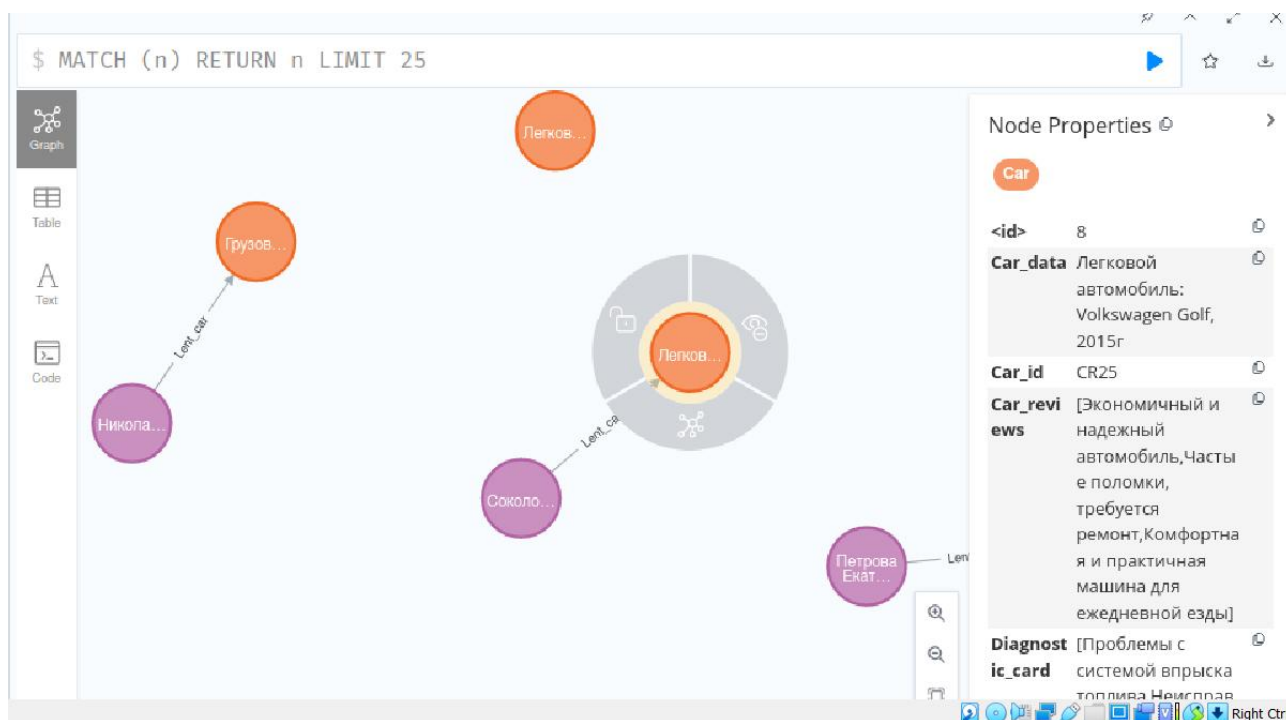


Рисунок 23 – Граф связей Арендатор-Арендовал-Автомобиль

Связи между узлами показаны серыми стрелками, и на рисунке 24 явно видна одна из таких связей. Справа на рисунке приведены поля, хранящиеся в этой связи: id аренды, дата начала аренды, количество дней, на которое был взят автомобиль и цена.

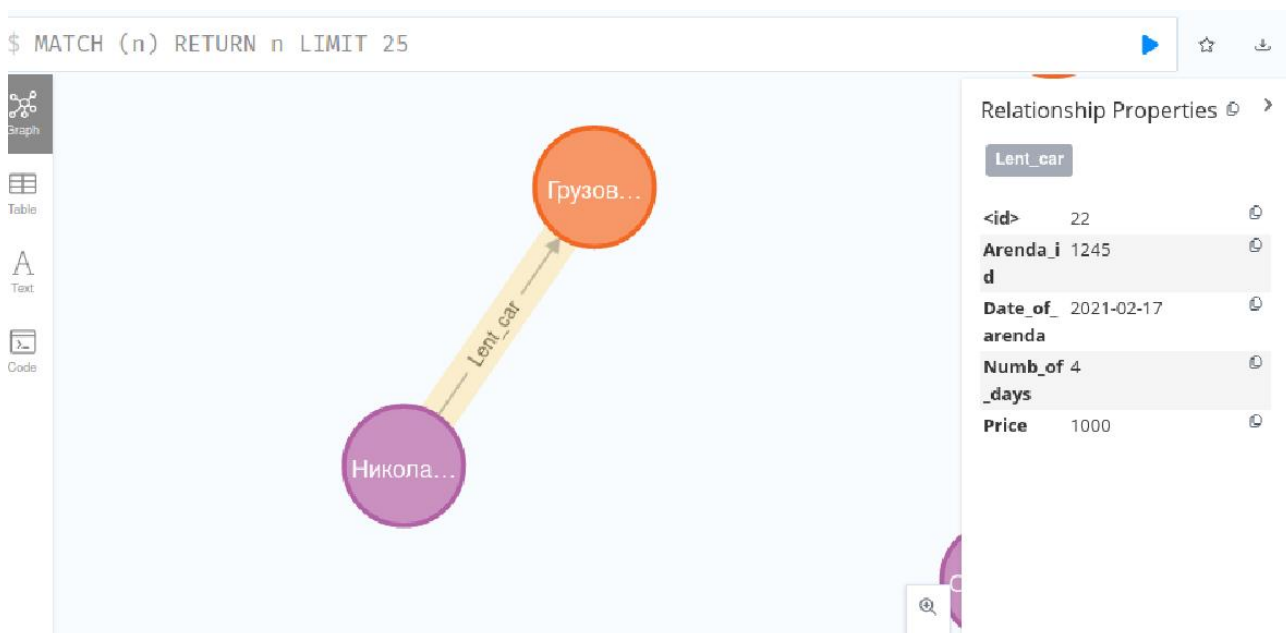


Рисунок 24 – Связь Арендатор-Арендовал-Автомобиль

### 3.4.2 Запрос к графовой базе данных

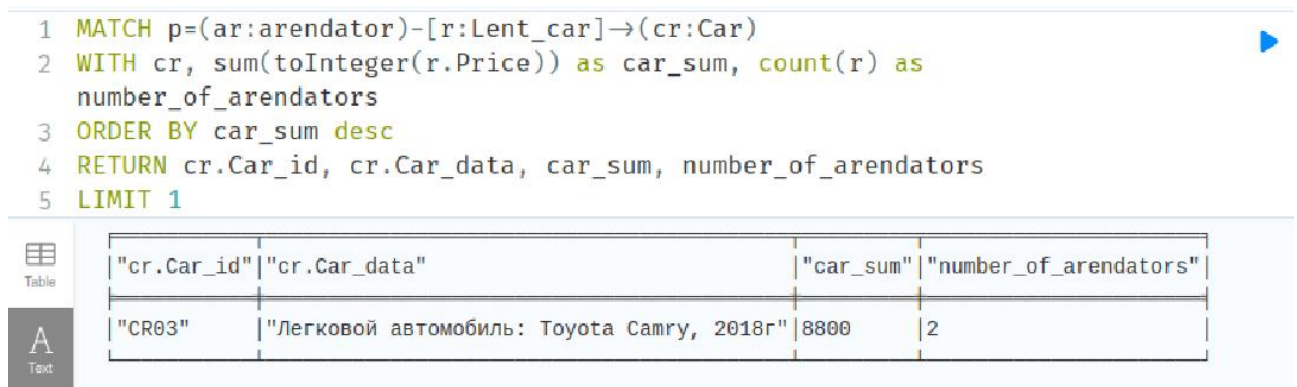
В соответствии с поставленной задачей требуется составить запрос: найти автомобиль с максимальной суммарной стоимостью аренды.

Запрос на языке Cypher:

```
MATCH p=(ar:arendator)-[r:Lent_car]->(cr:Car)
WITH cr, sum(toInteger(r.Price)) as car_sum, count(r) as
number_of_arendators
ORDER BY car_sum desc
RETURN cr.Car_id, cr.Car_data, car_sum, number_of_arendators
LIMIT 1
```

Этот запрос основан на связях типа "Арендатор-Арендовал-Автомобиль". Для каждой машины(cr), вычисляется общая стоимость аренды, указанная в каждой из этих связей. Результаты сортируются в порядке убывания (desc) суммарной стоимости аренды автомобиля, и оставляя только первую запись, мы получаем автомобиль с наибольшей общей стоимостью аренды.

На рисунке 25 показан результат выполнения этого запроса.



```
1 MATCH p=(ar:arendator)-[r:Lent_car]->(cr:Car)
2 WITH cr, sum(toInteger(r.Price)) as car_sum, count(r) as
  number_of_arendators
3 ORDER BY car_sum desc
4 RETURN cr.Car_id, cr.Car_data, car_sum, number_of_arendators
5 LIMIT 1
```

"cr.Car_id"	"cr.Car_data"	"car_sum"	"number_of_arendators"
"CR03"	"Легковой автомобиль: Toyota Camry, 2018г"	8800	2

Рисунок 25 – Результат выполнения запроса по поиску автомобиля с наибольшей суммарной стоимостью аренды

## **4. Работа со Spark**

### **4.1.Формирование таблиц**

В соответствии с заданием требуется сформировать csv-файлы (с внутренней схемой) таблиц «Арендатор», «Аренда», «Автомобиль» и сохранить их в файловой системе HDFS. Для выполнения данного задания была реализована программа с представленным ниже алгоритмом действий:

1. Установить соединение с кластером Elasticsearch, используя API.
2. Сделать подключение к сессии Spark.
3. Запросить и получить необходимые документы из ES с помощью запросов.
4. Создать схему данных, которая соответствует каждой таблице и задать соответствующие типы полей.
5. Заполнить таблицы, перебирая документы и преобразуя их поля в соответствии с заранее созданными схемами.
6. Создать датафреймы, используя схемы и полученные данные.
7. Записать датафреймы в CSV-файлы HDFS, настроив параметры, такие как заголовки столбцов и разрешение на перезапись.

На рисунке 26 проиллюстрирована схема алгоритма формирования таблиц и сохранения их в файловой системе HDFS. Код данной программы приведён в приложении Б.



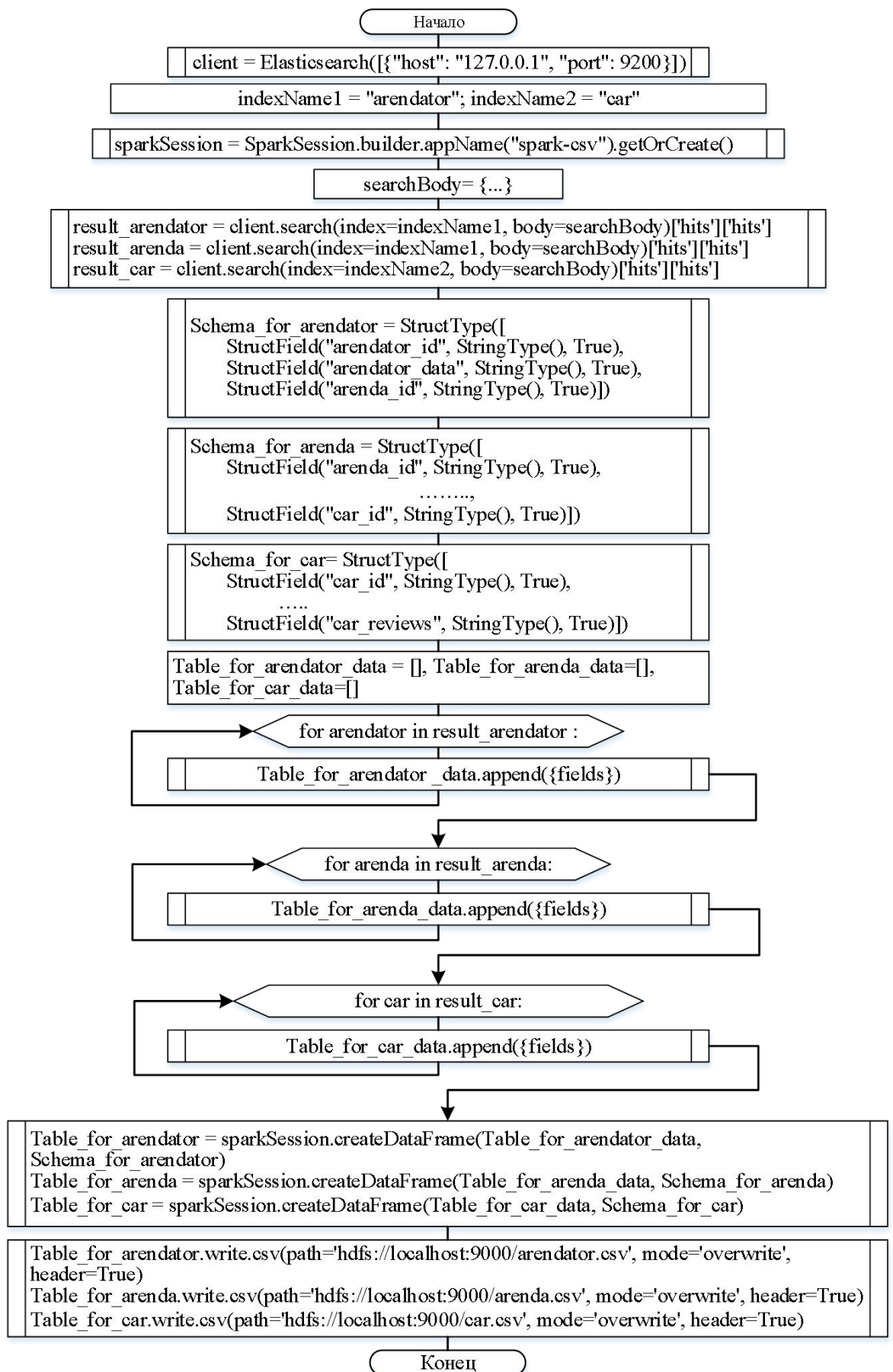


Рисунок 26 – Схема алгоритма формирования csv-файлов



После выполнения программы были созданы таблицы в соответствии с заданием. Представление этих таблиц приведено на рисунках 27, 28 и 29.

```

+-----+-----+-----+
|arendator_id|    arendator_data|arenda_id|
+-----+-----+-----+
|      AR01|Шаповалов Игорь М...|    1065|
|      AR02|Семенова Ева Влад...|    1089|
|      AR03|Иванов Александр ...|    1092|
|      AR04|Ковалева Анна Сер...|    1107|
|      AR05|Сидоров Иван Дмит...|    1123|
|      AR06|Петрова Екатерина...|    1156|
|      AR07|Григорьев Денис В...|    1187|
|      AR08|Антонов Максим Ал...|    1212|
|      AR09|Николаева Ольга И...|    1245|
|     AR10|Морозов Дмитрий С...|    1291|
|     AR11|Кузнецова Алексан...|    1352|
|     AR12|Смирнов Игорь Вас...|    1419|
|     AR13|Воронина Елена Ан...|    1494|
|     AR14|Герасимов Виктор ...|    1575|
|     AR15|Соколова Алена Па...|    1664|
|     AR16|Павлов Иван Викто...|    1761|
|     AR17|Козлова Ольга Вас...|    1866|
|     AR18|Ефимов Сергей Анд...|    1981|
|     AR19|Калинина Екатерин...|    2106|
|     AR20|Медведев Андрей И...|    2241|
+-----+-----+-----+
only showing top 20 rows

```

Рисунок 27 – Вывод файла arendator.csv

```

+-----+-----+-----+-----+-----+
|arenda_id|date_of_arena|numb_days_of_arend|price|car_id|
+-----+-----+-----+-----+-----+
|    1065|  2019-01-13|          10| 7000| CR03|
|    1089|  2020-03-23|           1|   500| CR12|
|    1092|  2020-03-26|           7|  1500| CR05|
|    1107|  2020-04-10|           5|  1200| CR15|
|    1123|  2020-05-02|           9|  2500| CR05|
|    1156|  2020-07-18|           3|   800| CR20|
|    1187|  2020-10-01|          12|  3500| CR11|
|    1212|  2020-12-03|           6|  1800| CR01|
|    1245|  2021-02-17|           4|  1000| CR28|
|    1291|  2021-06-08|           8|  2200| CR14|
|    1352|  2021-10-15|           3|   900| CR26|
|    1419|  2022-02-27|          11|  3200| CR09|
|    1494|  2022-07-10|           5|  1400| CR18|
|    1575|  2022-12-26|           9|  2500| CR06|
|    1664|  2023-05-12|           2|   600| CR25|
|    1761|  2023-10-27|           7|  1800| CR03|
|    1866|  2024-04-12|           6|  1500| CR19|
|    1981|  2024-10-01|           4|  1000| CR30|
|    2106|  2025-04-17|          10|  3000| CR13|
|    2241|  2025-11-05|           3|   900| CR23|
+-----+-----+-----+-----+-----+
only showing top 20 rows

```

Рисунок 28 – Вывод файла arenda.csv

car_id	car_data	diagnostic_card	car_reviews
CR01	Минивэн: Honda Od...	[Неисправность фа...	[Плохо заводится,...
CR02	Грузовик: Ford F-...	[Потертости на ку...	[Отличный вместит...
CR03	Легковой автомоби...	[Неисправность си...	[Комфортный и над...
CR04	Грузовик: Chevrol...	[Проблемы с транс...	[Надежный грузови...
CR05	Легковой автомоби...	[Проблемы с систе...	[Стильный и динам...
CR06	Грузовик: Ford Ra...	[Изношенные тормо...	[Мощный и надежны...
CR07	Легковой автомоби...	[Проблемы с систе...	[Роскошный и комф...
CR08	Грузовик: Toyota ...	[Потеря мощности ...	[Надежный грузови...
CR09	Легковой автомоби...	[Проблемы с систе...	[Надежный и эконо...
CR10	Грузовик: Ford Ra...	[Проблемы с систе...	[Надежный грузови...
CR11	Легковой автомоби...	[Неисправность си...	[Роскошный и комф...
CR12	Грузовик: Chevrol...	[Потертости на ку...	[Надежный и вмест...
CR13	Легковой автомоби...	[Проблемы с систе...	[Солидный и комфо...
CR14	Грузовик: GMC Sie...	[Изношенные тормо...	[Мощный и надежны...
CR15	Легковой автомоби...	[Проблемы с систе...	[Комфортный и сти...
CR16	Грузовик: Toyota ...	[Потеря мощности ...	[Надежный и мощны...
CR17	Легковой автомоби...	[Проблемы с систе...	[Экономичный и на...
CR18	Грузовик: Ford F-...	[Проблемы с систе...	[Надежный и вмест...
CR19	Легковой автомоби...	[Неисправность си...	[Комфортный и сти...
CR20	Грузовик: Chevrol...	[Изношенные тормо...	[Мощный и надежны...

only showing top 20 rows

Рисунок 29 – Вывод файла car.csv

## 4.2. Запрос в Spark

По заданию требуется реализовать запрос SELECT: определить арендатора и автомобиль с максимальной стоимостью аренды. На данный момент в Spark присутствуют следующие таблицы «arendator», «arenda» и «car».

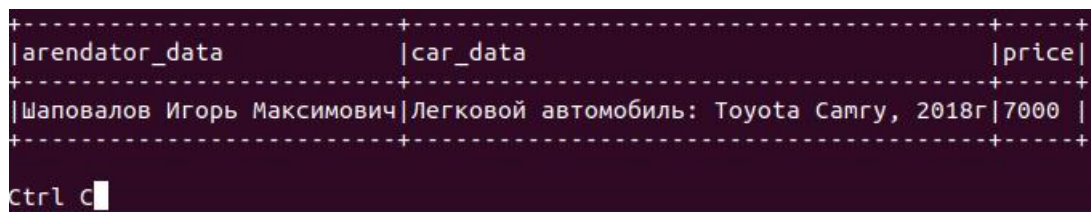
Реализуем запрос с помощью этих таблиц:

```
SELECT arendator.arendator_data, car.car_data, arenda.price
FROM arendator, arenda, car
WHERE (arenda.price = (SELECT MAX(price) FROM arenda)) and
      (arendator.arena_id = arenda.arena_id) and
      (arenda.car_id = car.car_id)
```

Для выполнения запроса была разработана Python-программа, которая функционирует следующим образом:

1. Начинается с установки соединения с сессией Spark.
2. Затем происходит чтение файлов arendator.csv, arenda.csv и car.csv.
3. Создаются временные таблицы на основе считанных данных.
4. После этого выполняется написанный выше SQL-запрос.

Код программы, необходимый для выполнения запроса, приведен в приложении Б, и результат этого запроса представлен на рисунке 30.



arendator_data	car_data	price
Шаповалов Игорь Максимович	Легковой автомобиль: Toyota Camry, 2018г	7000

ctrl c

Рисунок 30 – Результат выполнения запроса

### 4.3. Мониторинг Spark

Для мониторинга состояния и производительности задач в Spark было открыто окно веб-интерфейса, доступного по адресу <http://127.0.0.1:4040/jobs/>.

Когда программа с запросом запускается, Spark предоставляет информацию о временных характеристиках выполнения задач для данного скрипта. На рисунке 31 показаны все завершённые задачи.

▼ Completed Jobs (10)

Page: 1 1 Pages. Jump to 1 . Show 100 items in a page. Go

Job Id (Job Group) ▼	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
9	showString at NativeMethodAccessorImpl.java:0 showString at NativeMethodAccessorImpl.java:0	2023/09/10 19:41:38	0.2 s	1/1	1/1
8 (66cf410b-b85f-4fee-9c5e-642a638aa3c0)	broadcast exchange (runId 66cf410b-b85f-4fee-9c5e-642a638aa3c0) \$anonfun\$withThreadLocalCaptured\$1 at FutureTask.java:266	2023/09/10 19:41:38	0.2 s	1/1	1/1
7	\$anonfun\$withThreadLocalCaptured\$1 at FutureTask.java:266 \$anonfun\$withThreadLocalCaptured\$1 at FutureTask.java:266	2023/09/10 19:41:37	0.5 s	2/2	2/2
6 (76102693-b2e6-47ca-a70b-220011dc8b1a)	broadcast exchange (runId 76102693-b2e6-47ca-a70b-220011dc8b1a) \$anonfun\$withThreadLocalCaptured\$1 at FutureTask.java:266	2023/09/10 19:41:37	0.5 s	1/1	1/1
5	load at NativeMethodAccessorImpl.java:0 load at NativeMethodAccessorImpl.java:0	2023/09/10 19:41:35	77 ms	1/1	1/1
4	load at NativeMethodAccessorImpl.java:0 load at NativeMethodAccessorImpl.java:0	2023/09/10 19:41:35	42 ms	1/1	1/1
3	load at NativeMethodAccessorImpl.java:0 load at NativeMethodAccessorImpl.java:0	2023/09/10 19:41:35	91 ms	1/1	1/1
2	load at NativeMethodAccessorImpl.java:0 load at NativeMethodAccessorImpl.java:0	2023/09/10 19:41:35	55 ms	1/1	1/1
1	load at NativeMethodAccessorImpl.java:0 load at NativeMethodAccessorImpl.java:0	2023/09/10 19:41:34	0.3 s	1/1	1/1
0	load at NativeMethodAccessorImpl.java:0 load at NativeMethodAccessorImpl.java:0	2023/09/10 19:41:33	1 s	1/1	1/1

Рисунок 31 – Выполненные задачи скрипта

На рисунке 32 продемонстрирована временная диаграмма 6 задачи (job), которая непосредственно связана с SQL-запросом.

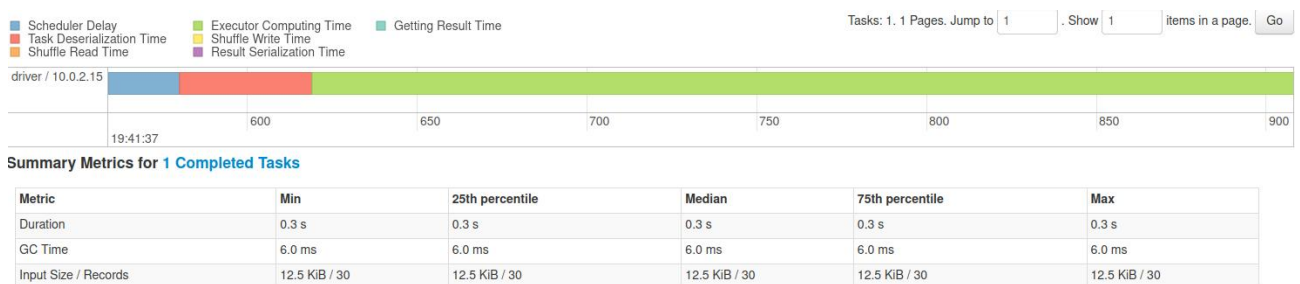


Рисунок 32 – Временная диаграмма задачи №6

Из временной диаграммы видно, что время выполнения запроса составило 0,3 секунды (Total Tasks Time), причем количество задач, которые были выполнены, зависело от количества разделов с записями (partitions). Чтобы уменьшить время выполнения запроса, было изменено число разделов с записями. В разделе SQL представлены данные о всех выполненных запросах. Скриншот этих задач показан на рисунке 33.



ID ▾	Description		Submitted	Duration	Job IDs
6	<a href="#">showString at NativeMethodAccessorImpl.java:0</a>	<a href="#">+details</a>	2023/09/10 19:41:37	2 s	<a href="#">[6]</a> <a href="#">[7]</a> <a href="#">[8]</a> <a href="#">[9]</a>
5	<a href="#">createOrReplaceTempView at NativeMethodAccessorImpl.java:0</a>	<a href="#">+details</a>	2023/09/10 19:41:36	4 ms	
4	<a href="#">createOrReplaceTempView at NativeMethodAccessorImpl.java:0</a>	<a href="#">+details</a>	2023/09/10 19:41:36	2 ms	
3	<a href="#">createOrReplaceTempView at NativeMethodAccessorImpl.java:0</a>	<a href="#">+details</a>	2023/09/10 19:41:36	5 ms	
2	<a href="#">load at NativeMethodAccessorImpl.java:0</a>	<a href="#">+details</a>	2023/09/10 19:41:35	0.1 s	<a href="#">[4]</a>
1	<a href="#">load at NativeMethodAccessorImpl.java:0</a>	<a href="#">+details</a>	2023/09/10 19:41:35	0.1 s	<a href="#">[2]</a>
0	<a href="#">load at NativeMethodAccessorImpl.java:0</a>	<a href="#">+details</a>	2023/09/10 19:41:31	3 s	<a href="#">[0]</a>

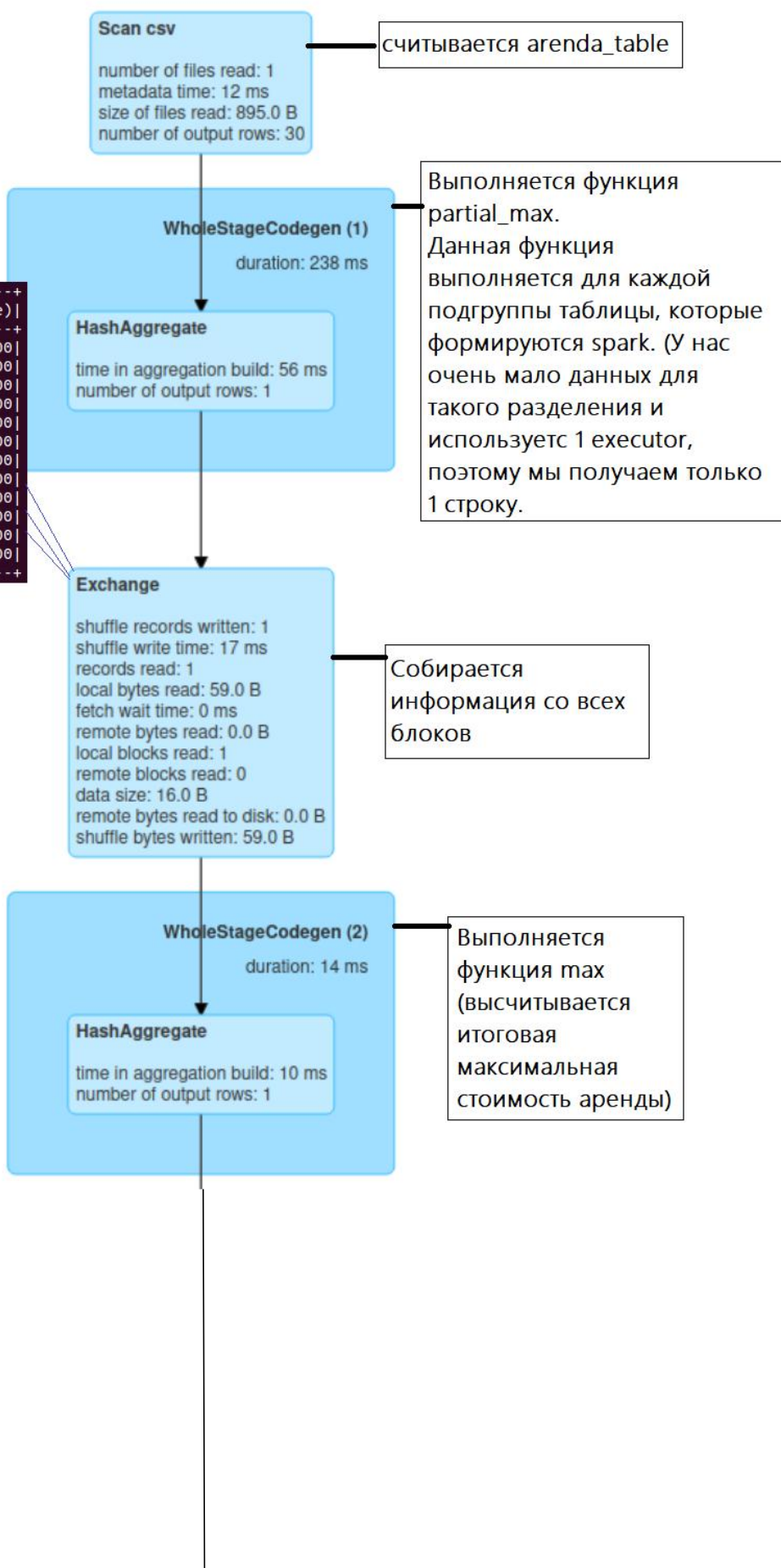
### Рисунок 33 – SQL-запросы сессии в Spark

Сосредотачиваясь на главном запросе с ID=6, давайте рассмотрим его более детально. Для наглядной иллюстрации стадий выполнения запроса и временных характеристик рассмотрим ориентированный ациклический граф (DAG). На рисунке 34 можно увидеть этот DAG, который отражает фазы выполнения основного SQL-запроса. Более подробный физический план выполнения запроса доступен в приложении Б.

Также на рисунке 34 даны комментарии к каждой ступени графа.

Если бы было  
разделение, то в  
каждом блоке  
получилось бы свое  
максимальное число,  
как в этой таблице

numb_days_of_arend	max(price)
12	3500
1	500
6	1800
3	900
5	1400
9	2500
4	1000
8	2200
7	1800
10	7000
11	3200
2	600



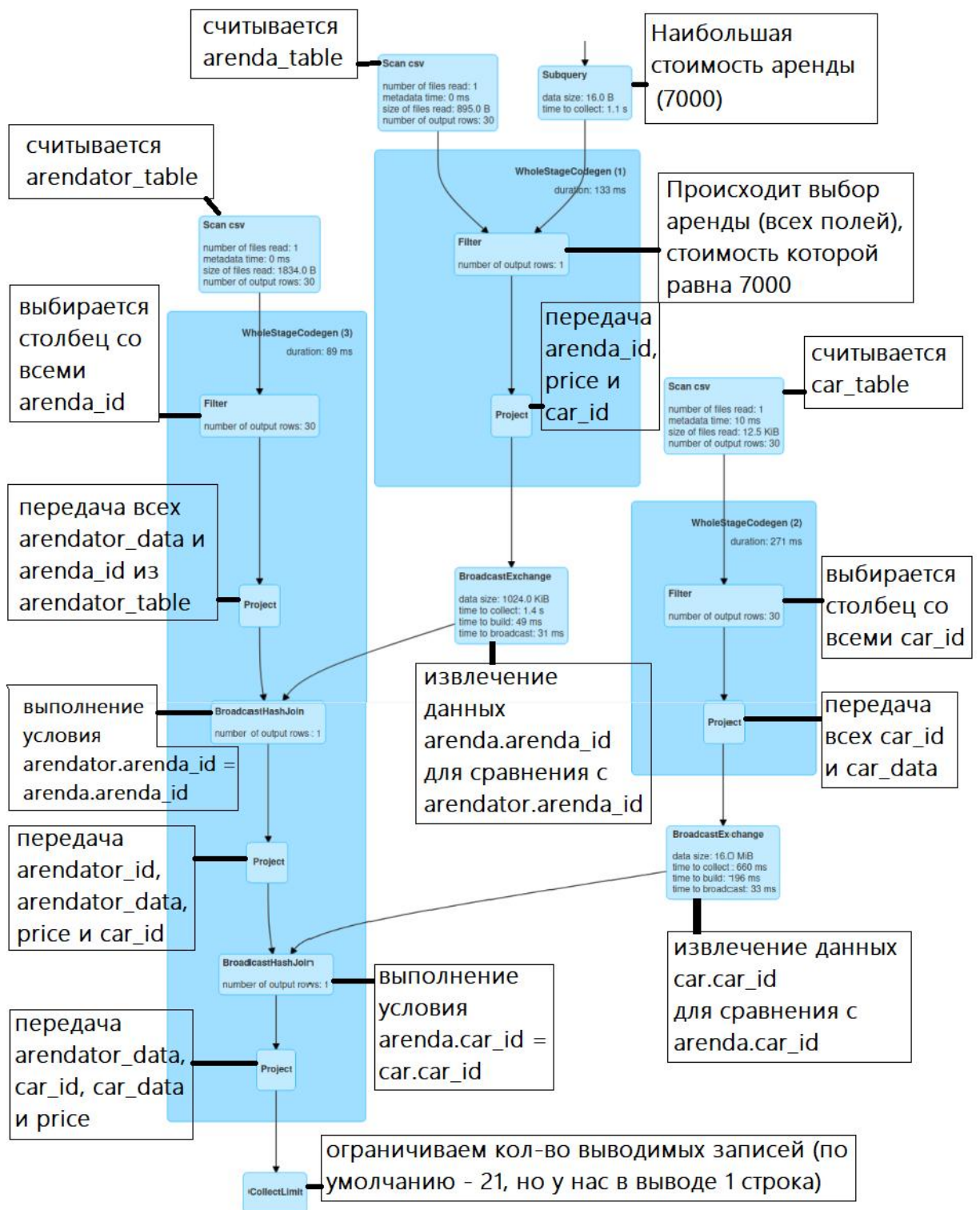


Рисунок 34 – DAG SQL-запроса

## ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта мы создали аналитическую систему аренды автомобилей, используя различные технологии и инструменты. Эта система строится на базе баз данных Elasticsearch и Neo4j, а также на платформах Hadoop и Spark. Мы установили необходимые программные компоненты на виртуальной машине Virtual Box, работающей под управлением операционной системы Ubuntu-20.

Одной из ключевых частей проекта была работа с данными, включая генерацию документов типа "Автомобиль" и "Арендатор". Мы создали кластер Elasticsearch с индексами Car и Arendator, а затем разработали программу маппинга с анализатором для русскоязычного текста. С использованием среды Kibana мы создали и выполнили запросы с вложенными агрегациями, что позволило нам анализировать данные более глубоко.

Дополнительно, мы создали и заполнили графовую базу данных Neo4j, в которой были созданы узлы "Арендатор" и "Автомобиль", а также определены связи между ними. Важной частью было представление связи "Арендатор-Арендовал-Автомобиль" в графическом виде, а также выполнение запроса на языке Cypher.

Чтобы работать с данными в Spark, мы создали CSV-файлы в файловой системе HDFS, содержащие таблицы "Аренда", "Арендатор" и "Автомобиль". Затем разработали SQL-запрос и проанализировали его с помощью монитора Spark.

В результате, наш проект представляет собой аналитическую систему аренды автомобилей, построенную на базах данных NoSQL, что позволяет проводить более глубокий анализ данных и может быть полезно для улучшения управления каршерингом и предоставления лучших услуг клиентам.



## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Neo4j – Графовая база данных. – URL: <https://neo4j.com/what-is-a-graph-database> (дата обращения 17.03.2023)
2. Hadoop: официальный сайт. – URL: <https://hadoop.apache.org/> (дата обращения 25.03.2023)
3. Apache Spark. – URL: <https://www.bigdataschool.ru/wiki/spark> (дата обращения 12.04.2023)
4. Elastic Matrix | Elasticsearch. – URL: [https://www.elastic.co/support/matrix#matrix\\_compatibility](https://www.elastic.co/support/matrix#matrix_compatibility) (дата обращения 17.03.2023)
5. Лабораторные работы 7-8 «Работа с Hadoop и Spark» по курсу «Технологии параллельных систем баз данных» – URL: <https://disk.yandex.ru/d/NkoTT8RvISqvnw> (дата обращения 20.04.2023)
6. Сайт Генератора ФИО - URL: <https://randomus.ru> (дата обращения 20.03.2022)
7. Онлайн-генератором JSON-документов URL: <https://app.json-generator.com/> (дата обращения 25.03.2023)
8. Официальная документация Elasticsearch – URL: <https://www.elastic.co/guide/en/elasticsearch/reference/current/getting-started-index.html> (дата обращения 27.04.2023).

## **ПРИЛОЖЕНИЕ А**

Созданные JSON-файлы и результаты выполнения запросов

## 1 Созданные JSON документы типа «Автомобиль»

```
[
  {
    "index": "car",
    "doc_type": "Car",
    "id": "CR01",
    "body": {
      "car_data": "Минивэн: Honda Odyssey, 2015г",
      "diagnostic_card": ["Неисправность фары", "Изношенные тормозные колодки"],
      "car_reviews": ["Плохо заводится", "Отличная машина, брала всего на пару часов, доехала быстро без проблем", "Машина супер, салон чистый, доехал быстро без проблем"]
    }
  },
  {
    "index": "car",
    "doc_type": "Car",
    "id": "CR02",
    "body": {
      "car_data": "Грузовик: Ford F-150, 2020г",
      "diagnostic_card": ["Потертости на кузове", "Рекомендуется заменить тормозные колодки в ближайшее время.", "Трещина на переднем стекле"],
      "car_reviews": ["Отличный вместительный кузов", "Не дорогая аренда авто, но машине не в лучшем состоянии", "Старенький грузовичок, но со своей задачей справляется"]
    }
  },
  {
    "index": "car",
    "doc_type": "Car",
    "id": "CR03",
    "body": {
      "car_data": "Легковой автомобиль: Toyota Camry, 2018г",
      "diagnostic_card": ["Неисправность системы кондиционирования", "Проблемы с электрикой"],
      "car_reviews": ["Комфортный и надежный автомобиль", "Редкие поломки, легко обслуживается"]
    }
  },
  {
    "index": "car",
    "doc_type": "Car",
    "id": "CR04",
    "body": {
      "car_data": "Грузовик: Chevrolet Silverado, 2017г",
      "diagnostic_card": ["Проблемы с трансмиссией", "Треск в рулевой колонке"],
      "car_reviews": ["Надежный грузовик для перевозки грузов", "Высокая проходимость в сложных условиях", "Просторный кузов"]
    }
  },
  {
    "index": "car",
    "doc_type": "Car",
    "id": "CR05",
    "body": {
      "car_data": "Легковой автомобиль: BMW 3 Series, 2019г",
      "diagnostic_card": ["Проблемы с системой охлаждения", "Стук в двигателе", "Шум в салоне", "Неисправность системы навигации"]
    }
  },
  {
    "index": "car",
    "doc_type": "Car",
    "id": "CR06",
    "body": {
      "car_data": "Грузовик: Ford Ranger, 2015г",
      "diagnostic_card": ["Изношенные тормозные диски", "Проблемы с системой выхлопа", "Течь масла"],
      "car_reviews": ["Мощный и надежный грузовик", "Некоторые проблемы с подвеской", "Просторная кабина для водителя и пассажиров"]
    }
  },
  {
    "index": "car",
    "doc_type": "Car",
    "id": "CR07",
    "body": {
      "car_data": "Легковой автомобиль: Mercedes-Benz E-Class, 2021г",
      "diagnostic_card": ["Проблемы с системой тормозов", "Неисправность системы стабилизации"]
    }
  }
]
```

```

        "car_reviews": ["Роскошный и комфортный автомобиль", "Редкие поломки, легко обслуживается",
"Мощный двигатель"]
    },
    {
        "index": "car",
        "doc_type": "Car",
        "id": "CR08",
        "body": {
            "car_data": "Грузовик: Toyota Tacoma, 2016г",
            "diagnostic_card": ["Потеря мощности двигателя", "Проблемы с трансмиссией"],
            "car_reviews": ["Надежный грузовик для перевозки грузов", "Хорошая проходимость в любых
условиях", "Просторная и удобная кабина"]
        }
    },
    {
        "index": "car",
        "doc_type": "Car",
        "id": "CR09",
        "body": {
            "car_data": "Легковой автомобиль: Volkswagen Golf, 2017г",
            "diagnostic_card": ["Проблемы с системой впрыска топлива", "Неисправность датчика давления в
шинах", "Треск в подвеске"],
            "car_reviews": ["Надежный и экономичный автомобиль", "Постоянные поломки, требуется ремонт",
"Удобная и практичная машина для городской езды", "Отличная устойчивость на дороге"]
        }
    },
    {
        "index": "car",
        "doc_type": "Car",
        "id": "CR10",
        "body": {
            "car_data": "Грузовик: Ford Ranger, 2011г",
            "diagnostic_card": ["Проблемы с системой охлаждения", "Шум в двигателе"],
            "car_reviews": ["Надежный грузов
ик для тяжелых нагрузок", "Проблемы с электрикой, требуется ремонт", "Хорошая проходимость в
бездорожье"]
        }
    },
    {
        "index": "car",
        "doc_type": "Car",
        "id": "CR11",
        "body": {
            "car_data": "Легковой автомобиль: Audi A4, 2020г",
            "diagnostic_card": ["Неисправность системы кондиционирования", "Проблемы с системой навигации",
"Треск в салоне"],
            "car_reviews": ["Роскошный и комфортный автомобиль", "Редкие поломки, легко обслуживается",
"Мощный двигатель и отличная управляемость"]
        }
    },
    {
        "index": "car",
        "doc_type": "Car",
        "id": "CR12",
        "body": {
            "car_data": "Грузовик: Chevrolet Silverado, 2014г",
            "diagnostic_card": ["Потертости на кузове", "Трещина на лобовом стекле", "Проблемы с системой
выхлопа"],
            "car_reviews": ["Надежный и вместительный грузовик", "Необходимы ремонтные работы",
"Просторный кузов для перевозки грузов"]
        }
    },
    {
        "index": "car",
        "doc_type": "Car",
        "id": "CR13",
        "body": {
            "car_data": "Легковой автомобиль: BMW X5, 2016г",
            "diagnostic_card": ["Проблемы с системой тормозов", "Шум в подвеске", "Неисправность системы
стабилизации"],
            "car_reviews": ["Солидный и комфортный автомобиль", "Периодически возникают поломки", "Мощный
двигатель и отличная устойчивость на дороге"]
        }
    },
    {
        "index": "car",
        "doc_type": "Car",
        "id": "CR14",

```

```

    "body": {
      "car_data": "Грузовик: GMC Sierra, 2019г",
      "diagnostic_card": ["Изношенные тормозные колодки", "Проблемы с системой выхлопа", "Течь
масла"],
      "car_reviews": ["Мощный и надежный грузовик", "Необходим ремонт", "Просторный кузов для
перевозки грузов"]
    }
  },
  {
    "index": "car",
    "doc_type": "Car",
    "id": "CR15",
    "body": {
      "car_data": "Легковой автомобиль: Mercedes-Benz C-Class, 2017г",
      "diagnostic_card": ["Проблемы с системой охлаждения", "Неисправность системы навигации"],
      "car_reviews": ["Комфортный и стильный автомобиль", "Редкие полом
ки, легко обслуживается", "Мощный двигатель и хорошая управляемость"]
    }
  },
  {
    "index": "car",
    "doc_type": "Car",
    "id": "CR16",
    "body": {
      "car_data": "Грузовик: Toyota Tundra, 2018г",
      "diagnostic_card": ["Потеря мощности двигателя", "Проблемы с трансмиссией", "Трещина на
переднем стекле"],
      "car_reviews": ["Надежный и мощный грузовик", "Хорошая проходимость в любых условиях",
"Просторная кабина и вместительный кузов"]
    }
  },
  {
    "index": "car",
    "doc_type": "Car",
    "id": "CR17",
    "body": {
      "car_data": "Легковой автомобиль: Volkswagen Passat, 2015г",
      "diagnostic_card": ["Проблемы с системой впрыска топлива", "Неисправность датчика давления в
шинах", "Треск в подвеске", "Стук в двигателе"],
      "car_reviews": ["Экономичный и надежный автомобиль", "Частые поломки, требуется ремонт",
"Комфортная и практичная машина для ежедневной езды"]
    }
  },
  {
    "index": "car",
    "doc_type": "Car",
    "id": "CR18",
    "body": {
      "car_data": "Грузовик: Ford F-150, 2013г",
      "diagnostic_card": ["Проблемы с системой охлаждения", "Шум в двигателе", "Течь масла",
"Потертости на кузове"],
      "car_reviews": ["Надежный и вместительный грузовик", "Требуется ремонт кузова", "Мощный
двигатель и хорошая проходимость"]
    }
  },
  {
    "index": "car",
    "doc_type": "Car",
    "id": "CR19",
    "body": {
      "car_data": "Легковой автомобиль: Audi Q5, 2022г",
      "diagnostic_card": ["Неисправность системы кондиционирования", "Проблемы с системой навигации",
"Треск в салоне", "Неисправность системы стабилизации"],
      "car_reviews": ["Комфортный и стильный автомобиль", "Редкие поломки, легко обслуживается",
"Мощный двигатель и отличная устойчивость на дороге"]
    }
  },
  {
    "index": "car",
    "doc_type": "Car",
    "id": "CR20",
    "body": {
      "car_data": "Грузовик: Chevrolet Colorado, 2021г",
      "diagnostic_card": ["Изношен
ные тормозные колодки", "Проблемы с системой выхлопа", "Течь масла", "Трещина на лобовом стекле"],
      "car_reviews": ["Мощный и надежный грузовик", "Необходим ремонт", "Просторный кузов для
перевозки грузов"]
    }
  }
}

```

```

},
{
  "index": "car",
  "doc_type": "Car",
  "id": "CR21",
  "body": {
    "car_data": "Легковой автомобиль: BMW 3 Series, 2019г",
    "diagnostic_card": ["Проблемы с системой тормозов", "Шум в подвеске", "Неисправность системы стабилизации", "Неисправность системы кондиционирования"],
    "car_reviews": ["Солидный и комфортный автомобиль", "Периодически возникают поломки", "Мощный двигатель и отличная управляемость"]
  }
},
{
  "index": "car",
  "doc_type": "Car",
  "id": "CR22",
  "body": {
    "car_data": "Грузовик: GMC Canyon, 2017г",
    "diagnostic_card": ["Потеря мощности двигателя", "Проблемы с трансмиссией", "Трещина на переднем стекле", "Потертости на кузове"],
    "car_reviews": ["Надежный и мощный грузовик", "Хорошая проходимость в любых условиях", "Просторная кабина и вместительный кузов"]
  }
},
{
  "index": "car",
  "doc_type": "Car",
  "id": "CR23",
  "body": {
    "car_data": "Легковой автомобиль: Mercedes-Benz GLC, 2016г",
    "diagnostic_card": ["Проблемы с системой охлаждения", "Неисправность системы навигации", "Треск в салоне"],
    "car_reviews": ["Комфортный и стильный автомобиль", "Редкие поломки, легко обслуживается", "Мощный двигатель и отличная устойчивость на дороге"]
  }
},
{
  "index": "car",
  "doc_type": "Car",
  "id": "CR24",
  "body": {
    "car_data": "Грузовик: Toyota Tundra, 2014г",
    "diagnostic_card": ["Изношенные тормозные колодки", "Проблемы с системой выхлопа", "Течь масла", "Потертости на кузове"],
    "car_reviews": ["Мощный и надежный грузовик", "Необходим ремонт", "Просторный кузов для перевозки грузов"]
  }
},
{
  "index": "car",
  "doc_type": "Car",
  "id": "CR25",
  "body": {
    "car_data": "Легковой автомобиль: Volkswagen Golf, 2015г",
    "diagnostic_card": ["Проблемы с системой впрыска топлива", "Неисправность датчика давления в шинах", "Треск в подвеске", "Стук в двигателе"],
    "car_reviews": ["Экономичный и надежный автомобиль", "Частые поломки, требуется ремонт", "Комфортная и практичная машина для ежедневной езды"]
  }
},
{
  "index": "car",
  "doc_type": "Car",
  "id": "CR26",
  "body": {
    "car_data": "Грузовик: Ford F-150, 2013г",
    "diagnostic_card": ["Проблемы с системой охлаждения", "Шум в двигателе", "Течь масла", "Потертости на кузове"],
    "car_reviews": ["Надежный и вместительный грузовик", "Требуется ремонт кузова", "Мощный двигатель и хорошая проходимость"]
  }
},
{
  "index": "car",
  "doc_type": "Car",
  "id": "CR27",
  "body": {

```

```

        "car_data": "Легковой автомобиль: Audi Q5, 2022г",
        "diagnostic_card": ["Неисправность системы кондиционирования", "Проблемы с системой навигации",
        "Треск в салоне", "Неисправность системы стабилизации"],
        "car_reviews": ["Комфортный и стильный автомобиль", "Редкие поломки, легко обслуживается",
        "Мощный двигатель и отличная устойчивость на дороге"]
    },
    {
        "index": "car",
        "doc_type": "Car",
        "id": "CR28",
        "body": {
            "car_data": "Грузовик: Chevrolet Colorado, 2021г",
            "diagnostic_card": ["Изношенные тормозные колодки", "Проблемы с системой выхлопа", "Течь
масла", "Трещина на лобовом стекле"],
            "car_reviews": ["Мощный и надежный грузовик", "Необходим ремонт", "Просторный кузов для
перевозки грузов"]
        }
    },
    {
        "index": "car",
        "doc_type": "Car",
        "id": "CR29",
        "body": {
            "car_data": "Легковой автомобиль: BMW 3 Series, 2019г",
            "diagnostic_card": ["Проблемы с системой тормозов", "Шум в подвеске", "Неисправность системы
стабилизации", "Неисправность системы кондиционирования"],
            "car_reviews": ["Солидный и ком
фортный автомобиль", "Периодически возникают поломки", "Мощный двигатель и отличная управляемость"]
        }
    },
    {
        "index": "car",
        "doc_type": "Car",
        "id": "CR30",
        "body": {
            "car_data": "Грузовик: GMC Canyon, 2017г",
            "diagnostic_card": ["Потеря мощности двигателя", "Проблемы с трансмиссией", "Трещина на
переднем стекле", "Потертости на кузове"],
            "car_reviews": ["Надежный и мощный грузовик", "Хорошая проходимость в любых условиях",
            "Просторная кабина и вместительный кузов"]
        }
    }
}

```

## 2 Созданные JSON документы типа «Арендатор»

```
[
{
  "index": "arendator",
  "doc_type": "Arendator",
  "id": "AR01",
  "body": {
    "arendator_id": "AR01",
    "arendator_data": "Шаповалов Игорь Максимович",
    "arenda_id": "1065",
    "date_of_arenda": "2019-01-13",
    "numb_days_of_arend": "10",
    "price": "7000",
    "car_id": "CR03"
  }
},
{
  "index": "arendator",
  "doc_type": "Arendator",
  "id": "AR02",
  "body": {
    "arendator_id": "AR02",
    "arendator_data": "Семенова Ева Владимировна",
    "arenda_id": "1089",
    "date_of_arenda": "2020-03-23",
    "numb_days_of_arend": "1",
    "price": "500",
    "car_id": "CR12"
  }
},
{
  "index": "arendator",
  "doc_type": "Arendator",
  "id": "AR03",
  "body": {
    "arendator_id": "AR03",
    "arendator_data": "Иванов Александр Петрович",
    "arenda_id": "1092",
    "date_of_arenda": "2020-03-26",
    "numb_days_of_arend": "7",
    "price": "1500",
    "car_id": "CR08"
  }
},
{
  "index": "arendator",
  "doc_type": "Arendator",
  "id": "AR04",
  "body": {
    "arendator_id": "AR04",
    "arendator_data": "Ковалева Анна Сергеевна",
    "arenda_id": "1107",
    "date_of_arenda": "2020-04-10",
    "numb_days_of_arend": "5",
    "price": "1200",
    "car_id": "CR15"
  }
},
{
  "index": "arendator",
  "doc_type": "Arendator",
  "id": "AR05",
  "body": {
    "arendator_id": "AR05",
    "arendator_data": "Сидоров Иван Дмитриевич",
    "arenda_id": "1123",
    "date_of_arenda": "2020-05-02",
    "numb_days_of_arend": "9",
    "price": "2500",
    "car_id": "CR05"
  }
},
{
  "index": "arendator",
  "doc_type": "Arendator",
  "id": "AR06",
  "body": {
```



```

        "arendator_id": "AR06",
        "arendator_data": "Петрова Екатерина Владимировна",
        "arenda_id": "1156",
        "date_of_arenda": "2020-07-18",
        "numb_days_of_arend": "3",
        "price": "800",
        "car_id": "CR20"
    }
},
{
    "index": "arendator",
    "doc_type": "Arendator",
    "id": "AR07",
    "body": {
        "arendator_id": "AR07",
        "arendator_data": "Григорьев Денис Викторович",
        "arenda_id": "1187",
        "date_of_arenda": "2020-10-01",
        "numb_days_of_arend": "12",
        "price": "3500",
        "car_id": "CR11"
    }
},
{
    "index": "arendator",
    "doc_type": "Arendator",
    "id": "AR08",
    "body": {
        "arendator_id": "AR08",
        "arendator_data": "Антонов Максим Александрович",
        "arenda_id": "1212",
        "date_of_arenda": "2020-12-03",
        "numb_days_of_arend": "6",
        "price": "1800",
        "car_id": "CR01"
    }
},
{
    "index": "arendator",
    "doc_type": "Arendator",
    "id": "AR09",
    "body": {
        "arendator_id": "AR09",
        "arendator_data": "Николаева Ольга Ивановна",
        "arenda_id": "1245",
        "date_of_arenda": "2021-02-17",
        "numb_days_of_arend": "4",
        "price": "1000",
        "car_id": "CR28"
    }
},
{
    "index": "arendator",
    "doc_type": "Arendator",
    "id": "AR10",
    "body": {
        "arendator_id": "AR10",
        "arendator_data": "Морозов Дмитрий Сергеевич",
        "arenda_id": "1291",
        "date_of_arenda": "2021-06-08",
        "numb_days_of_arend": "8",
        "price": "2200",
        "car_id": "CR14"
    }
},
{
    "index": "arendator",
    "doc_type": "Arendator",
    "id": "AR11",
    "body": {
        "arendator_id": "AR11",
        "arendator_data": "Кузнецова Александра Денисовна",
        "arenda_id": "1352",
        "date_of_arenda": "2021-10-15",
        "numb_days_of_arend": "3",
        "price": "900",
        "car_id": "CR26"
    }
}
}

```

```

},
{
  "index": "arendator",
  "doc_type": "Arendator",
  "id": "AR12",
  "body": {
    "arendator_id": "AR12",
    "arendator_data": "Смирнов Игорь Васильевич",
    "arenda_id": "1419",
    "date_of_arenda": "2022-02-27",
    "numb_days_of_arend": "11",
    "price": "3200",
    "car_id": "CR09"
  }
},
{
  "index": "arendator",
  "doc_type": "Arendator",
  "id": "AR13",
  "body": {
    "arendator_id": "AR13",
    "arendator_data": "Воронина Елена Андреевна",
    "arenda_id": "1494",
    "date_of_arenda": "2022-07
-10",
    "numb_days_of_arend": "5",
    "price": "1400",
    "car_id": "CR18"
  }
},
{
  "index": "arendator",
  "doc_type": "Arendator",
  "id": "AR14",
  "body": {
    "arendator_id": "AR14",
    "arendator_data": "Герасимов Виктор Михайлович",
    "arenda_id": "1575",
    "date_of_arenda": "2022-12-26",
    "numb_days_of_arend": "9",
    "price": "2500",
    "car_id": "CR06"
  }
},
{
  "index": "arendator",
  "doc_type": "Arendator",
  "id": "AR15",
  "body": {
    "arendator_id": "AR15",
    "arendator_data": "Соколова Алена Павловна",
    "arenda_id": "1664",
    "date_of_arenda": "2023-05-12",
    "numb_days_of_arend": "2",
    "price": "600",
    "car_id": "CR25"
  }
},
{
  "index": "arendator",
  "doc_type": "Arendator",
  "id": "AR16",
  "body": {
    "arendator_id": "AR16",
    "arendator_data": "Павлов Иван Викторович",
    "arenda_id": "1761",
    "date_of_arenda": "2023-10-27",
    "numb_days_of_arend": "7",
    "price": "1800",
    "car_id": "CR03"
  }
},
{
  "index": "arendator",
  "doc_type": "Arendator",
  "id": "AR17",
  "body": {
    "arendator_id": "AR17",
    "arendator_data": "Козлова Ольга Васильевна",

```

```

        "arenda_id": "1866",
        "date_of_arenda": "2024-04-12",
        "numb_days_of_arend": "6",
        "price": "1500",
        "car_id": "CR19"
    }
},
{
    "index": "arendator",
    "doc_type": "Arendator",
    "id": "AR18",
    "body": {
        "arendator_id": "AR18",
        "arendator_data": "Ефимов Сергей Андреевич",
        "arenda_id": "1981",
        "date_of_arenda": "2024-10-01",
        "numb_days_of_arend": "4",
        "price": "1000",
        "car_id": "CR30"
    }
},
{
    "index": "arendator",
    "doc_type": "Arendator",
    "id": "AR19",
    "body": {
        "arendator_id": "AR19",
        "arendator_data": "Калинина Екатерина Дмитриевна",
        "arenda_id": "2106",
        "date_of_arenda": "2025-04-17",
        "numb_days_of_arend": "10",
        "price": "3000",
        "car_id": "CR13"
    }
},
{
    "index": "arendator",
    "doc_type": "Arendator",
    "id": "AR20",
    "body": {
        "arendator_id": "AR20",
        "arendator_data": "Медведев Андрей Иванович",
        "arenda_id": "2241",
        "date_of_arenda": "2025-11-05",
        "numb_days_of_arend": "3",
        "price": "900",
        "car_id": "CR23"
    }
},
{
    "index": "arendator",
    "doc_type": "Arendator",
    "id": "AR21",
    "body": {
        "arendator_id": "AR21",
        "arendator_data": "Соловьева Ольга Петровна",
        "arenda_id": "2386",
        "date_of_arenda": "2026-06-25",
        "numb_days_of_arend": "8",
        "price": "2200",
        "car_id": "CR10"
    }
},
{
    "index": "arendator",
    "doc_type": "Arendator",
    "id": "AR22",
    "body": {
        "arendator_id": "AR22",
        "arendator_data": "Васильева Ирина Алексеевна",
        "arenda_id": "2541",
        "date_of_arenda": "2027-03-19",
        "numb_days_of_arend": "5",
        "price": "1400",
        "car_id": "CR21"
    }
},
{

```

```

    "index": "arendator",
    "doc_type": "Arendator",
    "id": "AR23",
    "body": {
      "arendator_id": "AR23",
      "arendator_data": "Кузьмин Дмитрий Михайлович",
      "arenda_id": "2716",
      "date_of_arenda": "2027-12-14",
      "numb_days_of_arend": "9",
      "price": "2500",
      "car_id": "CR07"
    }
  },
  {
    "index": "arendator",
    "doc_type": "Arendator",
    "id": "AR24",
    "body": {
      "arendator_id": "AR24",
      "arendator_data": "Михайлова Анастасия Владимировна",
      "arenda_id": "2911",
      "date_of_arenda": "2028-09-11",
      "numb_days_of_arend": "2",
      "price": "600",
      "car_id": "CR16"
    }
  },
  {
    "index": "arendator",
    "doc_type": "Arendator",
    "id": "AR25",
    "body": {
      "arendator_id": "AR25",
      "arendator_data": "Петрова Елена Александровна",
      "arenda_id": "3126",
      "date_of_arenda": "2029-06-11",
      "numb_days_of_arend": "7",
      "price": "1800",
      "car_id": "CR05"
    }
  },
  {
    "index": "arendator",
    "doc_type": "Arendator",
    "id": "AR26",
    "body": {
      "arendator_id": "AR26",
      "arendator_data": "Смирнов
а Ольга Дмитриевна",
      "arenda_id": "3361",
      "date_of_arenda": "2030-03-15",
      "numb_days_of_arend": "6",
      "price": "1500",
      "car_id": "CR22"
    }
  },
  {
    "index": "arendator",
    "doc_type": "Arendator",
    "id": "AR27",
    "body": {
      "arendator_id": "AR27",
      "arendator_data": "Козлов Василий Алексеевич",
      "arenda_id": "3616",
      "date_of_arenda": "2031-01-20",
      "numb_days_of_arend": "4",
      "price": "1000",
      "car_id": "CR29"
    }
  },
  {
    "index": "arendator",
    "doc_type": "Arendator",
    "id": "AR28",
    "body": {
      "arendator_id": "AR28",
      "arendator_data": "Соколова Анна Владимировна",
      "arenda_id": "3891",
      "date_of_arenda": "2031-11-26",

```

```

        "numb_days_of_arend": "10",
        "price": "3000",
        "car_id": "CR12"
    },
    {
        "index": "arendator",
        "doc_type": "Arendator",
        "id": "AR29",
        "body": {
            "arendator_id": "AR29",
            "arendator_data": "Егоров Алексей Иванович",
            "arenda_id": "4186",
            "date_of_arenda": "2032-10-04",
            "numb_days_of_arend": "3",
            "price": "900",
            "car_id": "CR24"
        }
    },
    {
        "index": "arendator",
        "doc_type": "Arendator",
        "id": "AR30",
        "body": {
            "arendator_id": "AR30",
            "arendator_data": "Максимова Елена Петровна",
            "arenda_id": "4501",
            "date_of_arenda": "2033-08-14",
            "numb_days_of_arend": "8",
            "price": "2200",
            "car_id": "CR11"
        }
    }
]

```

### 3 Результат выполнения первого запроса к кластеру ES

GET arendator/\_search

```
{
  "size": 0,
  "aggregations": {
    "year_period": {
      "date_histogram": {
        "field": "date_of_arenda",
        "calendar_interval": "year",
        "format": "yyyy-MM-dd"
      },
      "aggregations": {
        "Arendator": {
          "terms": {
            "field": "car_id",
            "order": {
              "_key": "asc"
            }
          },
          "aggregations": {
            "mean_arenda_price": {
              "avg": {
                "field": "price"
              }
            }
          }
        }
      }
    }
  }
}
```

---

```
{
  "took" : 58,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 30,
      "relation" : "eq"
    },
    "max_score" : null,
    "hits" : [ ]
  },
  "aggregations" : {
    "year_period" : {
      "buckets" : [
        {
          "key_as_string" : "2019-01-01",
          "key" : 1546300800000,
          "doc_count" : 1,
          "Arendator" : {
            "doc_count_error_upper_bound" : 0,
            "sum_other_doc_count" : 0,
            "buckets" : [
              {
                "key" : "cr03",
                "doc_count" : 1,

```

```

        "mean_arenda_price" : {
            "value" : 7000.0
        }
    }
]
}
},
{
    "key_as_string" : "2020-01-01",
    "key" : 1577836800000,
    "doc_count" : 7,
    "Arendator" : {
        "doc_count_error_upper_bound" : 0,
        "sum_other_doc_count" : 0,
        "buckets" : [
            {
                "key" : "cr01",
                "doc_count" : 1,
                "mean_arenda_price" : {
                    "value" : 1800.0
                }
            },
            {
                "key" : "cr05",
                "doc_count" : 2,
                "mean_arenda_price" : {
                    "value" : 2000.0
                }
            },
            {
                "key" : "cr11",
                "doc_count" : 1,
                "mean_arenda_price" : {
                    "value" : 3500.0
                }
            },
            {
                "key" : "cr12",
                "doc_count" : 1,
                "mean_arenda_price" : {
                    "value" : 500.0
                }
            },
            {
                "key" : "cr15",
                "doc_count" : 1,
                "mean_arenda_price" : {
                    "value" : 1200.0
                }
            },
            {
                "key" : "cr20",
                "doc_count" : 1,
                "mean_arenda_price" : {
                    "value" : 800.0
                }
            }
        ]
    }
}
},
{
    "key_as_string" : "2021-01-01",
    "key" : 1609459200000,
    "doc_count" : 3,

```

```

"Arendator" : {
  "doc_count_error_upper_bound" : 0,
  "sum_other_doc_count" : 0,
  "buckets" : [
    {
      "key" : "cr14",
      "doc_count" : 1,
      "mean_arena_price" : {
        "value" : 2200.0
      }
    },
    {
      "key" : "cr26",
      "doc_count" : 1,
      "mean_arena_price" : {
        "value" : 900.0
      }
    },
    {
      "key" : "cr28",
      "doc_count" : 1,
      "mean_arena_price" : {
        "value" : 1000.0
      }
    }
  ]
},
{
  "key_as_string" : "2022-01-01",
  "key" : 1640995200000,
  "doc_count" : 3,
  "Arendator" : {
    "doc_count_error_upper_bound" : 0,
    "sum_other_doc_count" : 0,
    "buckets" : [
      {
        "key" : "cr06",
        "doc_count" : 1,
        "mean_arena_price" : {
          "value" : 2500.0
        }
      },
      {
        "key" : "cr09",
        "doc_count" : 1,
        "mean_arena_price" : {
          "value" : 3200.0
        }
      },
      {
        "key" : "cr18",
        "doc_count" : 1,
        "mean_arena_price" : {
          "value" : 1400.0
        }
      }
    ]
  }
},
{
  "key_as_string" : "2023-01-01",
  "key" : 1672531200000,
  "doc_count" : 2,

```



```

    "Arendator" : {
      "doc_count_error_upper_bound" : 0,
      "sum_other_doc_count" : 0,
      "buckets" : [
        {
          "key" : "cr03",
          "doc_count" : 1,
          "mean_arenda_price" : {
            "value" : 1800.0
          }
        },
        {
          "key" : "cr25",
          "doc_count" : 1,
          "mean_arenda_price" : {
            "value" : 600.0
          }
        }
      ]
    }
  },
  {
    "key_as_string" : "2024-01-01",
    "key" : 1704067200000,
    "doc_count" : 2,
    "Arendator" : {
      "doc_count_error_upper_bound" : 0,
      "sum_other_doc_count" : 0,
      "buckets" : [
        {
          "key" : "cr19",
          "doc_count" : 1,
          "mean_arenda_price" : {
            "value" : 1500.0
          }
        },
        {
          "key" : "cr30",
          "doc_count" : 1,
          "mean_arenda_price" : {
            "value" : 1000.0
          }
        }
      ]
    }
  },
  {
    "key_as_string" : "2025-01-01",
    "key" : 1735689600000,
    "doc_count" : 2,
    "Arendator" : {
      "doc_count_error_upper_bound" : 0,
      "sum_other_doc_count" : 0,
      "buckets" : [
        {
          "key" : "cr13",
          "doc_count" : 1,
          "mean_arenda_price" : {
            "value" : 3000.0
          }
        },
        {
          "key" : "cr23",
          "doc_count" : 1,

```

```

        "mean_arenda_price" : {
            "value" : 900.0
        }
    }
]
}
},
{
    "key_as_string" : "2026-01-01",
    "key" : 1767225600000,
    "doc_count" : 1,
    "Arendator" : {
        "doc_count_error_upper_bound" : 0,
        "sum_other_doc_count" : 0,
        "buckets" : [
            {
                "key" : "cr10",
                "doc_count" : 1,
                "mean_arenda_price" : {
                    "value" : 2200.0
                }
            }
        ]
    }
},
{
    "key_as_string" : "2027-01-01",
    "key" : 1798761600000,
    "doc_count" : 2,
    "Arendator" : {
        "doc_count_error_upper_bound" : 0,
        "sum_other_doc_count" : 0,
        "buckets" : [
            {
                "key" : "cr07",
                "doc_count" : 1,
                "mean_arenda_price" : {
                    "value" : 2500.0
                }
            },
            {
                "key" : "cr21",
                "doc_count" : 1,
                "mean_arenda_price" : {
                    "value" : 1400.0
                }
            }
        ]
    }
},
{
    "key_as_string" : "2028-01-01",
    "key" : 1830297600000,
    "doc_count" : 1,
    "Arendator" : {
        "doc_count_error_upper_bound" : 0,
        "sum_other_doc_count" : 0,
        "buckets" : [
            {
                "key" : "cr16",
                "doc_count" : 1,
                "mean_arenda_price" : {
                    "value" : 600.0
                }
            }
        ]
    }
}

```

```

    }
  ]
}
},
{
  "key_as_string" : "2029-01-01",
  "key" : 18619200000000,
  "doc_count" : 1,
  "Arendator" : {
    "doc_count_error_upper_bound" : 0,
    "sum_other_doc_count" : 0,
    "buckets" : [
      {
        "key" : "cr05",
        "doc_count" : 1,
        "mean_arenda_price" : {
          "value" : 1800.0
        }
      }
    ]
  }
}
},
{
  "key_as_string" : "2030-01-01",
  "key" : 18934560000000,
  "doc_count" : 1,
  "Arendator" : {
    "doc_count_error_upper_bound" : 0,
    "sum_other_doc_count" : 0,
    "buckets" : [
      {
        "key" : "cr22",
        "doc_count" : 1,
        "mean_arenda_price" : {
          "value" : 1500.0
        }
      }
    ]
  }
}
},
{
  "key_as_string" : "2031-01-01",
  "key" : 19249920000000,
  "doc_count" : 2,
  "Arendator" : {
    "doc_count_error_upper_bound" : 0,
    "sum_other_doc_count" : 0,
    "buckets" : [
      {
        "key" : "cr12",
        "doc_count" : 1,
        "mean_arenda_price" : {
          "value" : 3000.0
        }
      },
      {
        "key" : "cr29",
        "doc_count" : 1,
        "mean_arenda_price" : {
          "value" : 1000.0
        }
      }
    ]
  }
}
}

```



```

{
  "took" : 45,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 15,
      "relation" : "eq"
    },
    "max_score" : 0.7457469,
    "hits" : [
      {
        "_index" : "car",
        "_type" : "Car",
        "_id" : "CR04",
        "_score" : 0.7457469,
        "_source" : {
          "car_data" : "Грузовик: Chevrolet Silverado, 2017г"
        }
      },
      {
        "_index" : "car",
        "_type" : "Car",
        "_id" : "CR06",
        "_score" : 0.7457469,
        "_source" : {
          "car_data" : "Грузовик: Ford Ranger, 2015г"
        }
      },
      {
        "_index" : "car",
        "_type" : "Car",
        "_id" : "CR08",
        "_score" : 0.7457469,
        "_source" : {
          "car_data" : "Грузовик: Toyota Tacoma, 2016г"
        }
      },
      {
        "_index" : "car",
        "_type" : "Car",
        "_id" : "CR10",
        "_score" : 0.7457469,
        "_source" : {
          "car_data" : "Грузовик: Ford Ranger, 2011г"
        }
      },
      {
        "_index" : "car",
        "_type" : "Car",
        "_id" : "CR12",
        "_score" : 0.7457469,
        "_source" : {
          "car_data" : "Грузовик: Chevrolet Silverado, 2014г"
        }
      },
      {
        "_index" : "car",
        "_type" : "Car",

```

```

    "_id" : "CR14",
    "_score" : 0.7457469,
    "_source" : {
      "car_data" : "Грузовик: GMC Sierra, 2019г"
    }
  },
  {
    "_index" : "car",
    "_type" : "Car",
    "_id" : "CR16",
    "_score" : 0.7457469,
    "_source" : {
      "car_data" : "Грузовик: Toyota Tundra, 2018г"
    }
  },
  {
    "_index" : "car",
    "_type" : "Car",
    "_id" : "CR20",
    "_score" : 0.7457469,
    "_source" : {
      "car_data" : "Грузовик: Chevrolet Colorado, 2021г"
    }
  },
  {
    "_index" : "car",
    "_type" : "Car",
    "_id" : "CR22",
    "_score" : 0.7457469,
    "_source" : {
      "car_data" : "Грузовик: GMC Canyon, 2017г"
    }
  },
  {
    "_index" : "car",
    "_type" : "Car",
    "_id" : "CR24",
    "_score" : 0.7457469,
    "_source" : {
      "car_data" : "Грузовик: Toyota Tundra, 2014г"
    }
  }
]
},
"aggregations" : {
  "count" : {
    "value" : 15
  }
}
}

```

## **ПРИЛОЖЕНИЕ Б**

Исходные коды программ

## 1 Программа маппинга и индексации документов типа «Автомобиль» и «Арендатор»

```
import json

from elasticsearch import Elasticsearch

client = Elasticsearch([{"host": "127.0.0.1", "port": 9200}])

index_1= "arendator"
index_2 = "car"

#client.indices.delete(index=index_1)
if client.indices.exists(index=index_1):
    print("Recreate " + index_1 + " index")
    client.indices.delete(index=index_1)
    client.indices.create(index=index_1)
else:
    print("Create " + index_1)
    client.indices.create(index=index_1)

if client.indices.exists(index=index_2):
    print("Recreate " + index_2 + " index")
    client.indices.delete(index=index_2)
    client.indices.create(index=index_2)
else:
    print("Create " + index_2)
    client.indices.create(index=index_2)

Procedure_Settings = {
    "analysis" : {
        "filter": {
            "russian_stop_words": {
                "type": "stop",
                "stopwords": "_russian_"
            },
            "filter_ru_sn": {
                "type": "snowball",
```



```

        "language": "Russian"
    }
},
"analyzer":
{
    "analitic_for_ru":
    {
        "type": "custom",
        "tokenizer": "standard",
        "filter": [
            "lowercase",
            "russian_stop_words",
            "filter_ru_sn"
        ]
    }
}
}
}

```

```

client.indices.close(index=index_1)
client.indices.put_settings(index=index_1, body=Procedure_Settings)
client.indices.close(index=index_2)
client.indices.put_settings(index=index_2, body=Procedure_Settings)

```

```

client.indices.open(index=index_1)

```

```

ArendatorMapping = {
    "properties": {
        "arendator_id": {
            "type": "text",
            "fielddata": True
        },
        "arendator_data": {
            "type": "text",
            "analyzer": "analitic_for_ru",
            "search_analyzer": "analitic_for_ru",

```

```

        "fielddata": True
    },
    "arenda_id": {
        "type": "text",
        "fielddata": True
    },
    "date_of_arenda": {
        "type": "date",
        "format": "yyyy-MM-dd"
    },
    "numb_days_of_arend": {
        "type": "integer"
    },
    "price": {
        "type": "integer"
    },
    "car_id": {
        "type": "text",
        "fielddata": True
    }
}
}

```

```

client.indices.put_mapping(index=index_1,
                           doc_type="Arendator",
                           include_type_name="true",
                           body=ArendatorMapping)

```

```

client.indices.open(index="arendator")

```

```

with open("/home/hadoopuser/js_files/Arendator.json", 'r') as file_Arendator:
    Arendator_data = json.load(file_Arendator)

```

```

for data in Arendator_data:

```

```

try:
    client.index(index=data["index"],
                 doc_type=data["doc_type"],
                 id=data["id"],
                 body=data["body"]
                 )
except Exception as e:
    print(e)
print("Arendator_indexed")

```

```

client.indices.open(index=index_2)

```

```

CarMapping = {
    "properties":{
        "car_data": {
            "type": "text",
            "analyzer":"analitic_for_ru",
            "search_analyzer":"analitic_for_ru",
            "fielddata": True
        },
        "diagnostic_card":{
            "type": "text",
            "analyzer":"analitic_for_ru",
            "search_analyzer":"analitic_for_ru",
            "fielddata": True
        },
        "car_reviews":{
            "type": "text",
            "analyzer":"analitic_for_ru",
            "search_analyzer":"analitic_for_ru",
            "fielddata": True
        }
    }
}

```

```

client.indices.put_mapping(index=index_2,

```

```

        doc_type="Car",
        include_type_name="true",
        body=CarMapping)

client.indices.open(index="car")

with open("/home/hadoopuser/js_files/Car.json", 'r') as file_Car:
    data_Car = json.load(file_Car)

for data in data_Car:
    try:
        client.index(index=data["index"],
                      doc_type=data["doc_type"],
                      id=data["id"],
                      body=data["body"]
                      )
    except Exception as e:
        print(e)
print("Car_indexed")

```

## 2 Тело первого запроса к кластеру ES

```
GET arendator/_search
{
  "size": 0,
  "aggregations": {
    "year_period": {
      "date_histogram": {
        "field": "date_of_arenda",
        "calendar_interval": "year",
        "format": "yyyy-MM-dd"
      },
      "aggregations": {
        "Arendator": {
          "terms": {
            "field": "car_id",
            "order": {
              "_key": "asc"
            }
          },
          "aggregations": {
            "mean_arend_a_price": {
              "avg": {
                "field": "price"
              }
            }
          }
        }
      }
    }
  }
}
```

---

## 3 Тело второго запроса к кластеру ES

```
GET car/_search
{
  "query": {
    "match_phrase": {
      "car_data": "грузовик"
    }
  },
  "_source": ["car_data"],
  "aggs": {
    "count": {
      "value_count": {
        "field": "_id"
      }
    }
  }
}
```

---

## 4 Импорт данных в базу данных Neo4j

```
from elasticsearch import Elasticsearch
from py2neo import Graph, Node, Relationship

client = Elasticsearch([{"host": "127.0.0.1", "port": 9200}])

indexName1 = "arendator"
indexName2 = "car"

graph_db = Graph("bolt://localhost:7687", auth=("neo4j", "sergey"))

graph_db.delete_all()

#array of arendator
arendator_array = {
    "size": 1000,
    "query": {
        "match_all": {}
    }
}

#array of cars
car_array = {
    "size": 1000,
    "query": {
        "match_all": {}
    }
}

result_arendator = client.search(index = indexName1, body =
arendator_array)
result_car = client.search(index = indexName2, body = car_array)

# cycle for my_node
for my_node in result_arendator ['hits']['hits']:
    Arendator_Node = Node("arendator",
        Arendator_id=my_node['_source']['arendator_id'],
        Arendator_data=my_node['_source']['arendator_data'])
```

```

graph_db.create(Arendator_Node)
try:
    for car in result_car ['hits']['hits']:
        Car_Node = graph_db.nodes.match("Car",
Car_id=car['_id']).first()
        if Car_Node == None:
            Car_Node = Node("Car", Car_id=car['_id'],
            Car_data = procedure['_source']['car_data'],

Diagnostic_card=procedure['_source']['diagnostic_card'],
            Car_reviews = procedure['_source']['car_reviews'])
graph_db.create(Car_Node)

# Link
if car['_id'] == my_node['_source']['car_id']:
    Lent_car = Relationship( Arendator_Node,
        "Lent_car", Car_Node,
        Arenda_id=my_node['_source']['arenda_id'],
        Date_of_arenda=my_node['_source']['date_of_arenda'],
        Numb_of_days=my_node['_source']['numb_days_of_arenda'],
        Price=my_node['_source']['price'])
graph_db.create(Lent_car)

except Exception as e:
    print(e)

```

## 5 Тело запроса запроса к базе данных Neo4j

```

MATCH p=(ar:arendator)-[r:Lent_car]->(cr:Car)
WITH cr, sum(toInteger(r.Price)) as car_sum, count(r) as
number_of_arendators
ORDER BY car_sum desc
RETURN cr.Car_id, cr.Car_data, car_sum, number_of_arendators
LIMIT 1

```

## 6 Программа создания CSV-файлов с таблицами

```
from elasticsearch import Elasticsearch
from pyspark.sql import SparkSession
from pyspark.sql.types import *
from datetime import datetime

client = Elasticsearch([{"host": "127.0.0.1", "port": 9200}])
indexName1 = "arendator"
indexName2 = "car"

sparkSession = SparkSession.builder.appName("spark-csv").getOrCreate()
searchBody = {
    "size": 60,
    "query": {
        "match_all": {}
    }
}

result_arendator = client.search(index=indexName1, body=searchBody)['hits']['hits']
result_arend = client.search(index=indexName1, body=searchBody)['hits']['hits']
result_car = client.search(index=indexName2, body=searchBody)['hits']['hits']

Schema_for_arendator = StructType([
    StructField("arendator_id", StringType(), True),
    StructField("arendator_data", StringType(), True),
    StructField("arenda_id", StringType(), True)
])

Schema_for_arend = StructType([
    StructField("arenda_id", StringType(), True),
    StructField("date_of_arend", StringType(), True),
    StructField("numb_days_of_arend", IntegerType(), True),
    StructField("price", IntegerType(), True),
    StructField("car_id", StringType(), True)
])

Schema_for_car = StructType([
    StructField("car_id", StringType(), True),
    StructField("car_data", StringType(), True),
    StructField("diagnostic_card", StringType(), True),
    StructField("car_reviews", StringType(), True)
])

# arendator table
Table_for_arendator_data = []
for arendator in result_arendator:
    Table_for_arendator_data.append((
        arendator['_source']['arendator_id'],
        arendator['_source']['arendator_data'],
        arendator['_source']['arenda_id']
    ))

# arenda table
Table_for_arend_data = []
for arenda in result_arend:
    Table_for_arend_data.append((
        arenda['_source']['arenda_id'],
        arenda['_source']['date_of_arend'],
        int(arend['_source']['numb_days_of_arend']),
        int(arend['_source']['price']),
        arenda['_source']['car_id']
    ))
```



```

# car table
Table_for_car_data = []
for car in result_car:
    Table_for_car_data.append((
        car['_id'],
        car['_source']['car_data'],
        car['_source']['diagnostic_card'],
        car['_source']['car_reviews']
    ))

#Creating data frame
Table_for_arendator = sparkSession.createDataFrame(Table_for_arendator_data, Schema_for_arendator)
Table_for_arenda = sparkSession.createDataFrame(Table_for_arenda_data, Schema_for_arenda)
Table_for_car = sparkSession.createDataFrame(Table_for_car_data, Schema_for_car)

# making csv
Table_for_arendator.write.csv(path='hdfs://localhost:9000/arendator.csv', mode='overwrite', header=True)
Table_for_arenda.write.csv(path='hdfs://localhost:9000/arenda.csv', mode='overwrite', header=True)
Table_for_car.write.csv(path='hdfs://localhost:9000/car.csv', mode='overwrite', header=True)

df_load = sparkSession.read.load(path='hdfs://localhost:9000/arendator.csv', format='csv', sep=',', inferSchema="true",
header="true")
df_load.show()
df_load = sparkSession.read.load(path='hdfs://localhost:9000/arenda.csv', format='csv', sep=',', inferSchema="true",
header="true")
df_load.show()
df_load = sparkSession.read.load(path='hdfs://localhost:9000/car.csv', format='csv', sep=',', inferSchema="true",
header="true")
df_load.show()

```

## 7 Программа запроса в Spark

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-from pyspark.sql import SparkSession

sparkSession=SparkSession.builder.appName("Python Spark SQL basic
example").config("spark.some.config.option", "5").getOrCreate()

Arendator_Table =
sparkSession.read.load(path='hdfs://localhost:9000/arendator.csv',
format='csv', sep=',', inferSchema="true", header="true")
Arenda_Table =
sparkSession.read.load(path='hdfs://localhost:9000/arenda.csv',
format='csv', sep=',', inferSchema="true", header="true")
Car_Table = sparkSession.read.load(path='hdfs://localhost:9000/car.csv',
format='csv', sep=',', inferSchema="true", header="true")

Arendator_Table.registerTempTable("arendator")
Arenda_Table.registerTempTable("arenda")
Car_Table.registerTempTable("car")

df = sparkSession.sql("
    SELECT arendator.arendator_id, arendator.arendator_data, car.car_id,
car.car_data, arenda.price
    FROM arendator, arenda, car
    WHERE (arenda.price = (SELECT MAX(price) FROM arenda))
    AND (arendator.arend_id = arenda.arend_id)
    AND (arenda.car_id = car.car_id)").show()

input('Ctrl C')
```

## 8 Физический план в мониторе Spark

```
== Parsed Logical Plan ==
GlobalLimit 21
+- LocalLimit 21
    +- Project [cast(arendator_id#16 as string) AS arendator_id#85,
cast(arendator_data#17 as string) AS arendator_data#86, cast(car_id#64 as string)
AS car_id#87, cast(car_data#65 as string) AS car_data#88, cast(price#41 as
string) AS price#89]
        +- Project [arendator_id#16, arendator_data#17, car_id#64, car_data#65,
price#41]
            +- Filter (((price#41 = scalar-subquery#72 []) AND (arenda_id#18 =
arenda_id#38)) AND (car_id#42 = car_id#64))
                : +- Aggregate [max(price#41) AS max(price)#74]
                : +- SubqueryAlias arenda
                : +-
Relation[arendator_id#16, arendator_data#17, arenda_id#18]
csv
    +- Join Inner
        :- Join Inner
        : :- SubqueryAlias arendator
        : : +- Relation[arendator_id#16, arendator_data#17, arenda_id#18]
csv
        : +- SubqueryAlias arenda
        : +-
Relation[arendator_id#16, arendator_data#17, arenda_id#18]
csv
    +- SubqueryAlias car
    +-
Relation[arendator_id#16, arendator_data#17, arenda_id#18]
csv

== Analyzed Logical Plan ==
arendator_id: string, arendator_data: string, car_id: string, car_data: string,
price: string
GlobalLimit 21
+- LocalLimit 21
    +- Project [cast(arendator_id#16 as string) AS arendator_id#85,
cast(arendator_data#17 as string) AS arendator_data#86, cast(car_id#64 as string)
AS car_id#87, cast(car_data#65 as string) AS car_data#88, cast(price#41 as
string) AS price#89]
        +- Project [arendator_id#16, arendator_data#17, car_id#64, car_data#65,
price#41]
            +- Filter (((price#41 = scalar-subquery#72 []) AND (arenda_id#18 =
arenda_id#38)) AND (car_id#42 = car_id#64))
                : +- Aggregate [max(price#41) AS max(price)#74]
                : +- SubqueryAlias arenda
                : +-
Relation[arendator_id#16, arendator_data#17, arenda_id#18]
csv
    +- Join Inner
        :- Join Inner
        : :- SubqueryAlias arendator
        : : +- Relation[arendator_id#16, arendator_data#17, arenda_id#18]
csv
        : +- SubqueryAlias arenda
        : +-
Relation[arendator_id#16, arendator_data#17, arenda_id#18]
csv
    +- SubqueryAlias car
    +-
Relation[arendator_id#16, arendator_data#17, arenda_id#18]
csv

== Optimized Logical Plan ==
GlobalLimit 21
```

```

+- LocalLimit 21
+- Project [arendator_id#16, arendator_data#17, car_id#64, car_data#65,
cast(price#41 as string) AS price#89]
+- Join Inner, (car_id#42 = car_id#64)
:- Project [arendator_id#16, arendator_data#17, price#41, car_id#42]
: +- Join Inner, (arenda_id#18 = arenda_id#38)
: :- Filter isnotnull(arenda_id#18)
: :- Relation[arendator_id#16,arendator_data#17,arenda_id#18]
csv
: +- Project [arenda_id#38, price#41, car_id#42]
: +- Filter (((isnotnull(price#41) AND (price#41 = scalar-
subquery#72 [])) AND isnotnull(arenda_id#38)) AND isnotnull(car_id#42))
: +- Aggregate [max(price#41) AS max(price)#74]
: +- Project [price#41]
: +-
Relation[arenda_id#38,date_of_arenda#39,numb_days_of_arend#40,price#41,car_id#42]
csv
: +-
Relation[arenda_id#38,date_of_arenda#39,numb_days_of_arend#40,price#41,car_id#42]
csv
+- Project [car_id#64, car_data#65]
+- Filter isnotnull(car_id#64)
+-
Relation[car_id#64,car_data#65,diagnostic_card#66,car_reviews#67] csv

== Physical Plan ==
CollectLimit 21
+- *(3) Project [arendator_id#16, arendator_data#17, car_id#64, car_data#65,
cast(price#41 as string) AS price#89]
+- *(3) BroadcastHashJoin [car_id#42], [car_id#64], Inner, BuildRight
:- *(3) Project [arendator_id#16, arendator_data#17, price#41, car_id#42]
: +- *(3) BroadcastHashJoin [arenda_id#18], [arenda_id#38], Inner,
BuildRight
: :- *(3) Project [arendator_id#16, arendator_data#17, arenda_id#18]
: :- *(3) Filter isnotnull(arenda_id#18)
: :- FileScan csv
[arendator_id#16,arendator_data#17,arenda_id#18] Batched: false, DataFilters:
[isnotnull(arenda_id#18)], Format: CSV, Location:
InMemoryFileIndex[hdfs://localhost:9000/arendator.csv], PartitionFilters: [],
PushedFilters: [IsNotNull(arenda_id)], ReadSchema:
struct<arendator_id:string,arendator_data:string,arenda_id:int>
: +- BroadcastExchange HashedRelationBroadcastMode(List(cast(input[0,
int, true] as bigint))), [id=#147]
: +- *(1) Project [arenda_id#38, price#41, car_id#42]
: +- *(1) Filter (((isnotnull(price#41) AND (price#41 = Subquery
scalar-subquery#72, [id=#124])) AND isnotnull(arenda_id#38)) AND
isnotnull(car_id#42))
: +- Subquery scalar-subquery#72, [id=#124]
: +- *(2) HashAggregate(keys=[],
functions=[max(price#41)], output=[max(price)#74])
: +- Exchange SinglePartition, true, [id=#120]
: +- *(1) HashAggregate(keys=[],
functions=[partial_max(price#41)], output=[max#96])
: +- FileScan csv [price#41] Batched: false,
DataFilters: [], Format: CSV, Location:
InMemoryFileIndex[hdfs://localhost:9000/arenda.csv], PartitionFilters: [],
PushedFilters: [], ReadSchema: struct<price:int>
: +- FileScan csv [arenda_id#38,price#41,car_id#42] Batched:
false, DataFilters: [isnotnull(price#41), isnotnull(arenda_id#38),
isnotnull(car_id#42)], Format: CSV, Location:
InMemoryFileIndex[hdfs://localhost:9000/arenda.csv], PartitionFilters: [],
PushedFilters: [IsNotNull(price), IsNotNull(arenda_id), IsNotNull(car_id)],
ReadSchema: struct<arenda_id:int,price:int,car_id:string>

```

```

      +- BroadcastExchange HashedRelationBroadcastMode(List(input[0, string,
true])), [id=#155]
      +- *(2) Project [car_id#64, car_data#65]
        +- *(2) Filter isnotnull(car_id#64)
          +- FileScan csv [car_id#64,car_data#65] Batched: false,
DataFilters: [isnotnull(car_id#64)], Format: CSV, Location:
InMemoryFileIndex[hdfs://localhost:9000/car.csv], PartitionFilters: [],
PushedFilters: [IsNotNull(car_id)], ReadSchema:
struct<car_id:string,car_data:string>

```

## **ПРИЛОЖЕНИЕ В**

Копии листов графической части

В графическую часть курсового проекта входят следующие листы:

- 1) Задание на курсовой проект;
- 2) Индексация документов Elasticsearch. Маппинг и анализатор;
- 3) Индексация документов Elasticsearch. Алгоритмы индексации;
- 4) Elasticsearch. Запросы;
- 5) Neo4j. Алгоритм создания и заполнения графовой БД;
- 6) Neo4j. Графическое представление данных и запрос;
- 7) Spark. Алгоритм создания CSV-файлов с таблицами;
- 8) Spark. Запрос;
- 9) Spark. Мониторинг выполнения запроса;
- 10) Spark. DAG SQL-запроса (1 часть);
- 11) Spark. DAG SQL-запроса (2 часть).