



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

# **РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

## ***К КУРСОВОМУ ПРОЕКТУ***

***НА ТЕМУ:***

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

***Санаторий***

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.

Дата выдачи задания « \_\_\_\_ » \_\_\_\_\_ 20\_\_ г.

## РЕФЕРАТ

РПЗ 90 с., 34 рис., 1 табл., 8 источн., 3 прил.

САНАТОРИЙ, ПАЦИЕНТЫ, ПРОЦЕДУРА, АНАЛИТИЧЕСКАЯ СИСТЕМА, БАЗА ДАННЫХ, БОЛЬШИЕ ДАННЫЕ, ПАРАЛЛЕЛЬНЫЕ СИСТЕМЫ, NOSQL, ELASTICSEARCH, NEO4J, SPARK, HADOOP, HDFS, KIBANA, ВИЗУАЛИЗАЦИЯ ДАННЫХ.

Курсовой проект направлен на разработку макета аналитической системы для санатория с применением NoSQL-баз данных, таких как Elasticsearch, Neo4j, Hadoop и Spark. Реализация проекта включала в себя следующие шаги:

- Установка виртуальной машины с Ubuntu-20.04.3 в VirtualBox и настройка необходимых программных компонентов;
- Установка систем NoSQL, Elasticsearch, Neo4j, Hadoop и Spark;
- Генерация JSON-файлов, содержащих информацию о пациентах и процедурах, для заполнения базы данных;
- Создание индекса с маппингом и анализатором, а также индексация JSON-документов;
- Разработка и выполнение запросов к данным в соответствии с поставленной задачей;
- Визуализация данных с использованием среды Kibana;
- Заполнение графовой базы данных и выполнение запросов в ней;
- Анализ выполнения запросов с помощью монитора.

## СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ .....	6
ВВЕДЕНИЕ .....	3
1. Задание и описание варианта курсового проекта .....	6
2. Установка используемых средств .....	8
2.1. Установка виртуальной машины .....	8
2.2. Установка Elasticsearch и Kibana .....	10
2.3. Установка Neo4j .....	13
2.4. Установка Hadoop и Spark .....	14
3. Работа с Elasticsearch .....	17
3.1. Создание JSON-документов .....	17
3.2. Индексация документов .....	18
3.3. Запросы с вложенной агрегацией к данным ES .....	24
3.3.1. Первый запрос .....	24
3.3.2. Второй запрос .....	27
3.4 Neo4j .....	29
3.4.1 Создание и заполнение графовой базы данных .....	29
3.4.2 Запрос к графовой базе данных .....	32
4. Spark .....	33
4.1. Создание и заполнение таблиц .....	33
4.2. Запрос в Spark .....	36
4.3. Мониторинг Spark .....	38
ЗАКЛЮЧЕНИЕ .....	41
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....	42
ПРИЛОЖЕНИЕ А .....	43
ПРИЛОЖЕНИЕ Б .....	67
ПРИЛОЖЕНИЕ В .....	79

## ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

ОС – операционная система

ОП – оперативная память

ВМ – виртуальная машина

БД – база данных

СУБД – система управления базой данных

DAG (Directed Acyclic Graph) - это направленный ациклический граф, в котором вершины представляют задачи или операции, а ребра - зависимости между ними

ELK (Elasticsearch, Logstash, Kibana) – стек технологий, используемый для сбора, обработки и анализа журналов и метрик данных

ES – Elasticsearch

DDB – распределенная база данных

HDFS (Hadoop Distributed File System) – распределённая файловая система для хранения больших объёмов данных на кластерах серверов

MapReduce – модель распределенных вычислений, используемая для параллельных вычислений над очень большими объемами данных

Стемминг – это процесс приведения слова к его основе (или корню) путем удаления суффиксов и окончаний

Токенизация – процесс разбиения текста на фрагменты, которые называются токенами

Хост-машина – машина, на которой запускается другая виртуальная операционная система

Кластер (ES) – группа из одного или нескольких связанных узлов Elasticsearch

Маппинг – это процесс определения того, как поля документа будут храниться и индексироваться

Kibana – это свободно распространяемая программа для визуализации и анализа данных в Elasticsearch

## ВВЕДЕНИЕ

В настоящее время, компьютеры основного класса уступают место параллельным системам баз данных во многих ИТ-проектах. Параллельные СУБД позволяют работать с гораздо большими объемами данных, обеспечивают высокую доступность и производительность серверов, при этом имеют более привлекательную цену. В процессе выполнения курсового проекта использовались технологии параллельных СУБД, которые остаются востребованными в свете описанных выше преимуществ.

Курсовой проект направлен на создание прототипа аналитической системы для санатория, с основной целью обработки и анализа информации о пациентах и процедурах. Для реализации данного проекта были использованы современные NoSQL технологии, включая Elasticsearch, Hadoop, Spark и Neo4j.

Система собирает данные о процедурах и пациентах, представленных в различных форматах в зависимости от выбранной базы данных. Для этой цели использовались Elasticsearch и графовая база данных Neo4j. Дополнительно, в проекте используются csv-файлы, которые хранятся в файловой системе HDFS.

Elasticsearch - это распределенная система управления поисковыми запросами и аналитикой, которая работает на основе открытого кода Apache Lucene. Elasticsearch использует JSON-подобный документный стиль для хранения и индексации данных. Он позволяет выполнять быстрый и точный поиск по большим объемам данных, а также предоставляет мощные инструменты аналитики и визуализации данных. Elasticsearch может интегрироваться с различными приложениями и инструментами, такими как Kibana, Logstash и Beats, для управления, анализа и визуализации данных.

Neo4j - это графовая база данных, которая использует графовую модель данных для хранения и обработки информации. Она основана на языке запросов Cypher, который позволяет легко и гибко извлекать данные из графовой структуры. База данных Neo4j широко используется для решения задач, связанных с обработкой и анализом связанных данных, таких как социальные сети, геоданные, рекомендательные системы и т.д. Она также

обладает высокой производительностью и масштабируемостью, что делает ее популярным выбором для обработки больших объемов данных.

Neo4j - это графовая база данных, которая использует графовую модель данных для хранения и обработки информации. Она основана на языке запросов Cypher, который позволяет легко и гибко извлекать данные из графовой структуры. База данных Neo4j широко используется для решения задач, связанных с обработкой и анализом связанных данных, таких как социальные сети, геоданные, рекомендательные системы и т.д. Она также обладает высокой производительностью и масштабируемостью, что делает ее популярным выбором для обработки больших объемов данных [1].

Apache Hadoop - это фреймворк с открытым исходным кодом, который позволяет распределенно обрабатывать и хранить большие объемы данных на кластере из нескольких компьютеров [2]. Hadoop основан на модели MapReduce, которая позволяет разбивать большие задачи на более мелкие и распределять их между узлами кластера для параллельной обработки. Он также включает в себя распределенную файловую систему HDFS, которая предоставляет масштабируемый и отказоустойчивый способ хранения данных на кластере. Hadoop широко используется для обработки больших объемов данных в различных областях, включая банковское дело, медицину, науку, социальные сети и другие.

Для управления множеством файлов на различных узлах используется распределенная файловая система HDFS (Hadoop Distributed File System). В HDFS файлы хранятся на различных серверах (DataNodes), а их местоположение управляется сервером имен (NameNode). Благодаря такой организации HDFS может хранить файлы значительного размера.

Для обеспечения надежности, масштабируемости и правильной работы распределенных приложений важно строго следить за соответствием версий продуктов и правильно устанавливать пакеты. Для этой задачи в проекте использовался пакетный менеджер YARN (Yet Another Resource Negotiator). Помимо этого, при работе с Hadoop важно упомянуть о фреймворке Spark,

основном применяемом для обработки больших данных. Spark используется для распределенной пакетной и потоковой обработки неструктурированных и слабоструктурированных данных в Hadoop [3].

Apache Spark - это фреймворк с открытым исходным кодом для обработки больших объемов данных, который использует распределенные вычисления в памяти для обеспечения высокой производительности. Он широко используется для анализа данных, машинного обучения и потоковой обработки данных. Spark обеспечивает интеграцию с Hadoop и другими системами, что делает его универсальным инструментом для работы с большими данными.

В ходе выполнения курсового проекта были применены различные технологии и продукты. Для реализации аналитического макета санатория были выполнены следующие этапы:

1. Установка виртуализационного программного продукта VirtualBox для операционной системы Linux;
2. Установка операционной системы Ubuntu 20.04.3 на виртуальную машину;
3. Установка необходимых программных пакетов, включая Elasticsearch, Neo4j, Hadoop и Spark;
4. Создание JSON-файлов с исходными данными;
5. Создание индексов в Elasticsearch с использованием анализатора и маппинга;
6. Реализация запросов к БД Elasticsearch и визуализация результатов в Kibana;
7. Заполнение графовой базы данных Neo4j данными из Elasticsearch;
8. Реализация запросов к БД Neo4j;
9. Создание таблиц в формате CSV и сохранение их в распределенной файловой системе HDFS;
10. Реализация запросов в Apache Spark;
11. Мониторинг процесса выполнения запросов в Apache Spark.



## 1. Задание и описание варианта курсового проекта

Для успешной реализации курсового проекта необходимо выполнить несколько задач.

1. Необходимо установить виртуальную машину с ОС Ubuntu 20.04.3 в VirtualBox с определенными характеристиками, такими как объем оперативной памяти и диск.

2. Установить Elasticsearch, Neo4j, Hadoop и Spark.

3. Автоматически создать два JSON-файла с 20-30 JSON-документами каждого типа для предметной области "Санаторий" с указанными полями документов.

4. Для индексации JSON-документов в Elasticsearch необходимо создать индекс с анализатором и маппингом, а затем проиндексировать документы.

5. Разработать запросы с вложенной агрегацией и представить результаты в Kibana.

6. В Neo4j необходимо заполнить графовую базу данных данными из Elasticsearch, разработать и реализовать запрос к этой БД.

7. В Spark нужно сформировать csv-файлы с таблицами и сохранить их в файловой системе HDFS по данным из Elasticsearch.

8. Необходимо написать и реализовать запрос в Spark, а затем проанализировать процесс выполнения запроса с помощью мониторинга.

В 24 варианте предметной областью курсового проекта является Санаторий, для которого необходимо сгенерировать два JSON-документа с определенными полями:

Пациент:

```
{index, doc_type, id, body: {id_пациента, персональные_данные*, путёвка, дата_прибытия, продолжительность_прибывания, [диагноз*], [id_процедуры]}}
```

Процедура:

```
{index, doc_type, id, body: {сведения_о_процедуре*, стоимость}}
```

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 пациентов по дате прибытия с периодом 1 год, для каждой «корзины» определить количество пациентов, прошедших каждую процедуру;
- определить стоимость процедуры по заданным ключевым словам.

Результаты данных запросов представить в среде Kibana.

Для графовой БД Neo4j необходимо сделать следующее:

1. По данным из Elasticsearch заполнить графовую базу данных Пациент(id\_пациента, дата\_прибытия, персональные\_данные) - Прошёл(стоимость) - Процедура(id\_процедуры).  
Примечание. В скобках приведены свойства узлов и отношения (связи), глагол – это отношение. Между двумя узлами может быть несколько связей одного типа.
2. Разработать и реализовать запрос: найти процедуру с максимальной суммарной стоимостью.

Для Spark необходимо выполнить следующие задачи:

1. По данным из Elasticsearch сформировать csv-файлы (с внутренней схемой) таблиц «Пациент», «Назначение», «Процедура» и сохранить их в файловой системе HDFS.
2. Написать запрос select: определить суммарное число назначений по каждой процедуре.
3. Реализовать этот запрос в Spark. Построить временную диаграмму его выполнения по результатам работы монитора.

## 2. Установка используемых средств

### 2.1. Установка виртуальной машины

В начале работы над курсовым проектом была установлена VirtualBox - программа виртуализации для операционной системы Linux. Эта среда обладает несколькими преимуществами, такими как компактность, высокая производительность и возможность бесплатного использования.

В VirtualBox была создана виртуальная машина с требуемыми характеристиками, в соответствии с заданием. Для установки использовался дистрибутив Ubuntu 20.30.40 - ubuntu-20.04.3-desktop-amd64.iso.

Процесс установки виртуальной машины Ubuntu с операционной системой Ubuntu 20.30.40 представлена на рисунке 1.

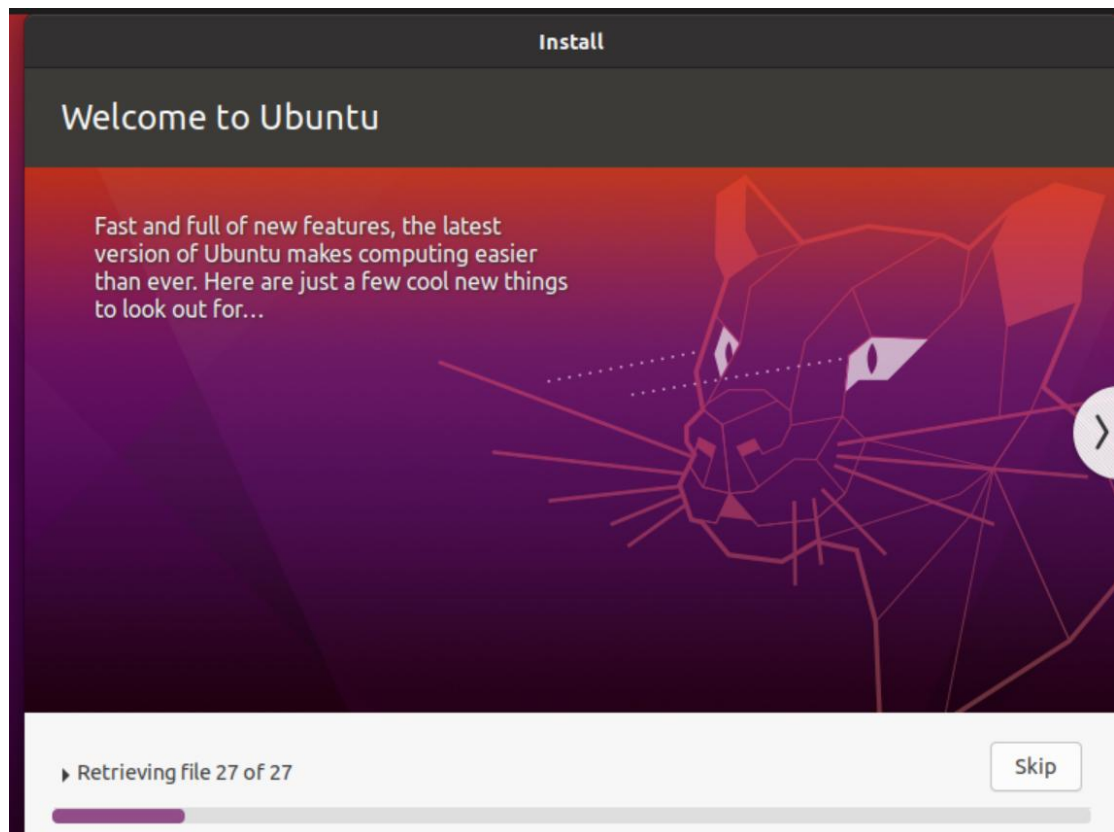


Рисунок 1 – Конфигурация VM Ubuntu-01

После успешной установки Ubuntu 20.30.40 в VirtualBox была создана учетная запись с логином "hadoopuser" и паролем "hadoop". Для облегчения работы в данной среде были установлены функции общего буфера и Drag'n'Drop, которые позволяют копировать текст и команды между основной операционной системой и виртуальной машиной в VirtualBox. Обмен между

ними был настроен в двух направлениях, что позволяет использовать эти функции в обе стороны.

На рисунке 2 приведен пример настроек данных функций.

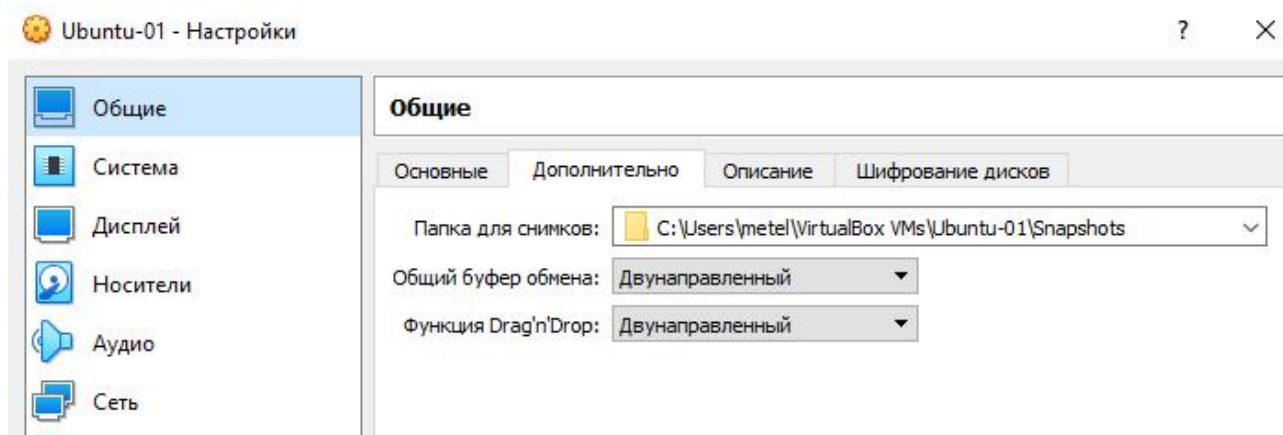


Рисунок 2 – Функции Drag'n'Drop для VirtualBox

## 2.2. Установка Elasticsearch и Kibana

Для продолжения работы было необходимо установить Elasticsearch. Этот продукт имеет значительные преимущества, такие как хранение больших объемов данных, быстрый поиск по ним и возможность анализа данных практически в режиме реального времени. Elasticsearch использует библиотеку Lucene для выполнения поисковых запросов, что делает ее доступной для различных платформ. Рекомендуется установка Java не ниже 8 версии, так как Lucene реализована на языке Java. На рисунке 3 показан результат выполнения команды «java -version», который подтверждает, что установлена версия Java 11.0.15, соответствующая требованиям.

```
openjdk version "11.0.15" 2022-04-19
OpenJDK Runtime Environment (build 11.0.15+10-Ubuntu-0ubuntu0.20.04.1)
OpenJDK 64-Bit Server VM (build 11.0.15+10-Ubuntu-0ubuntu0.20.04.1, mixed mode,
sharing)
```

Рисунок 3 – Результат выполнения команды java -version

Для проверки работоспособности сервера Elasticsearch необходимо выполнить команду "curl -X GET localhost:9200". Результат выполнения команды представлен на рисунке 4.

```
anton@anton-VirtualBox:~$ sudo systemctl enable elasticsearch
Synchronizing state of elasticsearch.service with SysV service s
cript with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable elasticsearch
Created symlink /etc/systemd/system/multi-user.target.wants/elasticsearch.service → /lib/systemd/system/elasticsearch.service.
anton@anton-VirtualBox:~$ curl -X GET 'http://localhost:9200'
{
  "name" : "anton-VirtualBox",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "i-xihelsQR6km5aAVWY6RQ",
  "version" : {
    "number" : "7.10.1",
    "build_flavor" : "default",
    "build_type" : "deb",
    "build_hash" : "1c34507e66d7db1211f66f3513706fdf548736aa",
    "build_date" : "2020-12-05T01:00:33.671820Z",
    "build_snapshot" : false,
    "lucene_version" : "8.7.0",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```

Рисунок 4 – Вывод команды «curl -X GET "localhost:9200"»

Elasticsearch предоставляет такие возможности, как хранение больших объемов данных, быстрый поиск и анализ данных в режиме реального времени. Библиотека Lucene, реализованная на языке Java, используется в Elasticsearch для выполнения поисковых запросов.

ES 7.10.1 находится в активном состоянии, как показывает вывод команды. Дополнительно, мы можем узнать версию ES, кластер и имя ВМ. ES можно рассматривать как базу данных NoSQL, поскольку неструктурированные данные хранятся в JSON-файлах, однако ES также позволяет выполнять поисковые запросы и ряд других функций.

Хранимые в ES файлы могут быть просмотрены в браузере, и для этой цели наша работа использует Mozilla. В Mozilla файлы JSON отображаются более наглядно благодаря подсветке и возможности свернуть отдельные части документа. На рисунке 5 представлен пример начальной страницы ES в браузере Mozilla.

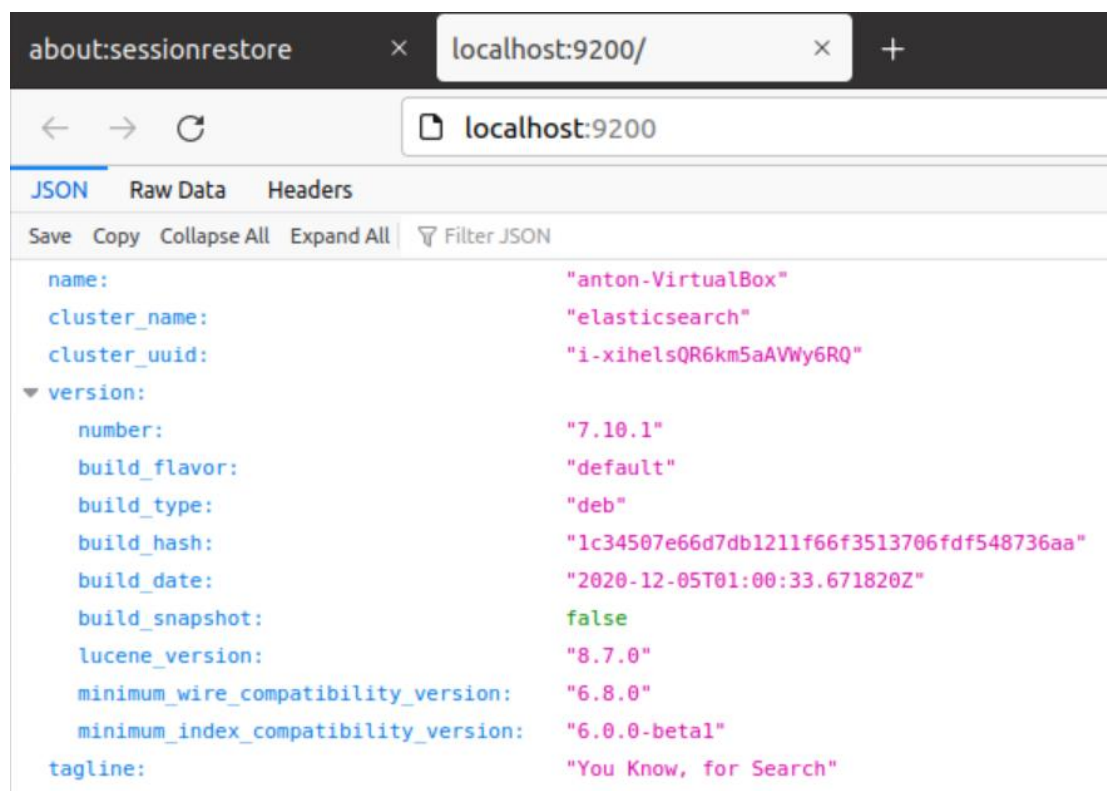


Рисунок 5 – Начальная страница ES

Для работы с Elasticsearch используется среда Kibana. Установка Kibana производится при помощи команды «`sudo apt-get install kibana=x`», где `x` -

версия Kibana. Стоит отметить, что для каждой версии Elasticsearch необходима определенная версия Kibana. Совместимость версий можно проверить на официальном сайте [4]. В данной работе была установлена версия 7.10.1 Kibana, поскольку она совместима с Elasticsearch версии 7.10.1. После запуска сервиса была проверена работоспособность данного сервиса, для этого в браузере был сделан переход по адресу <http://localhost:5601/app/kibana>. На рисунке 6 показан веб-интерфейс данного сервиса.

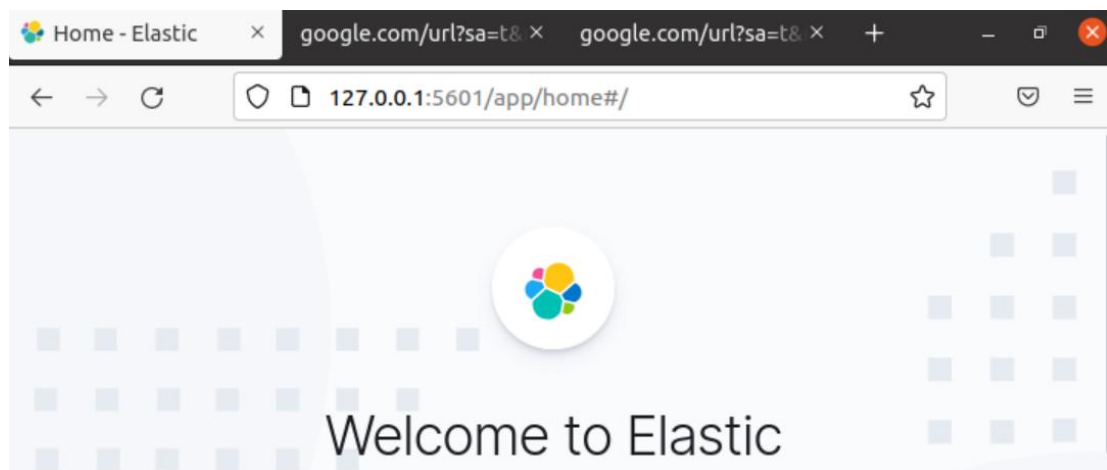


Рисунок 6 – Web-интерфейс Kibana

Веб-интерфейс Kibana предоставляет пользователю доступ к данным, хранящимся в Elasticsearch. В Kibana можно создавать дашборды, визуализации, делать поисковые запросы и анализировать данные. Один из преимуществ Kibana - это возможность визуального анализа данных, который значительно упрощает их интерпретацию. На рисунке 7 видно, что веб-интерфейс Kibana имеет панель навигации, где пользователь может выбрать интересующую его визуализацию или дашборд, а также меню поиска и выбора индексов.



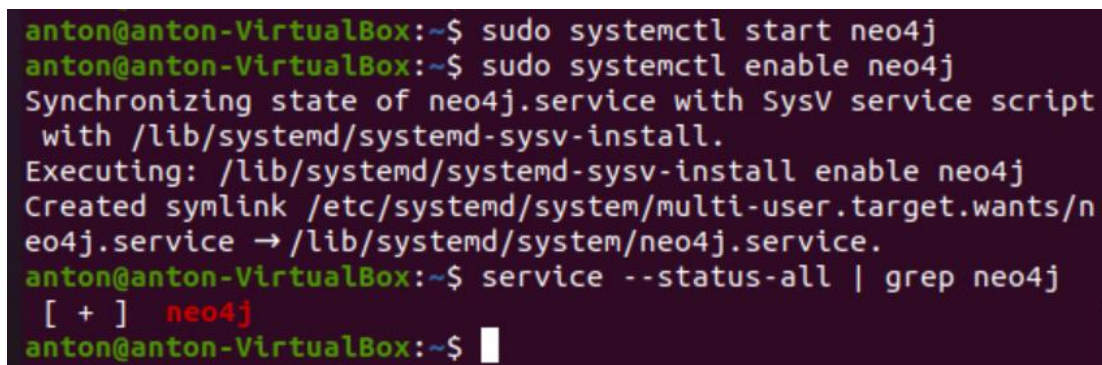
Рисунок 7 – Панель навигации Kibana



### 2.3. Установка Neo4j

Для установки Neo4j были выполнены несколько шагов. Сначала был импортирован ключ GPG, затем был добавлен сам пакет Neo4j к менеджеру пакетов apt. После этого были обновлены индексы пакетов, а затем установлен сервис при помощи команды «sudo apt install neo4j=3.5.14».

После установки Neo4j был запущен сам сервис командой «sudo systemctl start neo4j», а также был активирован автоматический запуск при каждом включении ВМ с помощью команды «sudo systemctl enable neo4j». Для проверки работоспособности графовой СУБД была использована команда «service --status-all | grep neo4j», результатом которой является успешная установка и запуск сервиса. Детали выполненных шагов можно посмотреть на рисунке 8.



```
anton@anton-VirtualBox:~$ sudo systemctl start neo4j
anton@anton-VirtualBox:~$ sudo systemctl enable neo4j
Synchronizing state of neo4j.service with SysV service script
with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable neo4j
Created symlink /etc/systemd/system/multi-user.target.wants/n
eo4j.service -> /lib/systemd/system/neo4j.service.
anton@anton-VirtualBox:~$ service --status-all | grep neo4j
[ + ] neo4j
anton@anton-VirtualBox:~$
```

Рисунок 8 – Статус установки и запуска Neo4j

Также для Neo4j имеется графический web-интерфейс, доступный по адресу <http://localhost:7474>.



## 2.4. Установка Hadoop и Spark

Для установки Hadoop и Spark использовались рекомендации и инструкции, представленные в методических указаниях лабораторных работ 7-8 по курсу «Технология параллельных систем баз данных».

Сначала были загружены архивы для установки - `hadoop-3.3.0.tar.gz` и `spark-3.2.1-bin-hadoop2.7.tgz`. Затем была создана группа `hadoop` и в ней был создан пользователь `hadoopuser` для работы с Hadoop. После этого архивы были распакованы, и в файлах конфигурации были внесены необходимые исправления для правильной работы системы [5]. Результат работы команд представлен на рисунке 9.

```
hadoopuser@anton-VirtualBox:~$ wget https://downloads.apache.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 9 - Скачивание ключа для архива файлов Надоор-установки

Далее был отформатирован узел Nadoor, что представлено на рисунке 10.

```
hadoopuser@anton-VirtualBox:~$ hdfs namenode -format
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:/u
(terminated)
```

Рисунок 10 - Форматирование узла Nadoor

Также для корректной работы Nadoop был запущен сервер SSH и сформирован ключ rsa. Результат генерации ключа представлен на рисунке 11.

```

hadoopuser@anton-VirtualBox:~$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hadoopuser/.ssh/id_rsa):
Created directory '/home/hadoopuser/.ssh'.
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:nG+a/L3zAh/Avq6drrfDb4s0qLB4X4YgzuowQ2zEwes hadoopuser@anton-VirtualBox
The key's randomart image is:
+---[RSA 3072]-----+
|..                    |
|...                   |
| o.                   |
|o.      . .o         |
|.+. .   S. .         |
|oE . . . oo .        |
|+ o . . +.=+ .       |
|. + ..o = *===       |
|o.....o *B0==*.     |
+---[SHA256]-----+
hadoopuser@anton-VirtualBox:~$

```

Рисунок 11 - Генерация ключа RSA

Таким образом, был создан однонодовый кластер Hadoop.

Для запуска Hadoop используются менеджеры HDFS и YARN. Для запуска HDFS применяется команда «start-dfs.sh», а для запуска YARN - «start-yarn.sh». Но для того чтобы не вводить отдельно все команды был создан скрипт для быстрого запуска. Результат работы скрипта показан на рисунке 12.

```

hadoopuser@anton-VirtualBox:~$ sudo chmod 777 hdp.sh
hadoopuser@anton-VirtualBox:~$ ./hdp.sh
Starting namenodes on [localhost]
localhost: namenode is running as process 2273. Stop it first.
Starting datanodes
localhost: datanode is running as process 2406. Stop it first.
Starting secondary namenodes [anton-VirtualBox]
anton-VirtualBox: secondarynamenode is running as process 2591. Stop it first.
Starting resourcemanager
resourcemanager is running as process 2796. Stop it first.
Starting nodemanagers
localhost: nodemanager is running as process 2920. Stop it first.
hadoopuser@anton-VirtualBox:~$

```

Рисунок 12 - Вывод скрипта по запуску процессов Hadoop

После запуска Nadoor начинают работать различные процессы и демоны, которые могут быть просмотрены при помощи команды «jprs». Пример вывода этой команды представлен на рисунке 13.

```
hadoopuser@anton-VirtualBox:~$ jps
2273 NameNode
3492 Jps
2406 DataNode
2920 NodeManager
2796 ResourceManager
2591 SecondaryNameNode
hadoopuser@anton-VirtualBox:~$ s
```

Рисунок 13 – Запущенные процессы и демоны Nadoop

Для успешной установки Spark были внесены корректировки в `.bashrc`. Подтверждение установки Spark представлено на рисунке 14.

[illegible]

### Рисунок 14 – Результат запуска Spark

### 3. Работа с Elasticsearch

#### 3.1.Создание JSON-документов

Для загрузки необходимых данных в кластер Elasticsearch сначала необходимо создать два JSON-документа: «Пациент» и «Процедура». Для их создания использовались различные онлайн-сервисы. Вначале был создан список, содержащий фамилии, имена и отчества, при помощи генератора ФИО[6]. В итоге были созданы две программы, которые были обработаны онлайн-генератором JSON-документов [7].

В результате были созданы два файла: Procedure.json, содержащий 30 записей, и Patient.json, содержащий также 30 записей. Оба файла можно просмотреть в Приложении А.

В документе типа Процедура содержатся следующие данные:

- name – название процедуры, представлено строкой от одного до трех слов;
- description – подробное описание процедуры с описанием используемых средств, а также результата, которого может ожидать пациент после данной процедуры;
- price – цена данной процедуры (в рублях).

При генерации файла Пациент уникальные идентификаторы процедур были взяты из того же списка, который использовался для заполнения файла Процедура.

В документе типа Пациент содержатся следующие данные:

- patient\_id – уникальный набор символов и цифр, позволяющий идентифицировать пациента санатория;
- personal\_data – ФИО пациента;
- voucher – уникальный набор семи цифр, позволяющий идентифицировать номер путевки, выданный пациенту санатория;
- date\_of\_arrival – дата прибытия пациента в санаторий;
- number\_of\_days – количество дней, проведенных пациентом в санатории;



- diagnosis – перечень диагнозов пациента;
- precedures\_id – перечень id-процедур, которые были назначены и пройдены пациентом санатория.

### 3.2. Индексация документов

Elasticsearch (ES) - это база данных, ориентированная на хранение документов, где каждый документ содержит все необходимые данные и должен быть проиндексирован. При таком подходе производительность высока, но изменение данных может быть затруднительным. Индексация документов в Elasticsearch заключается в добавлении их в индексы.

Для организации анализа текста используется маппинг - создание схемы данных, которая определяет поля документов, которые будут анализироваться в дальнейшем. В работе применяется анализатор, который обеспечивает улучшенный полнотекстовый поиск по документам [8].

Анализатор состоит из трех основных компонентов: символьного фильтра, токенизатора и фильтра токенов.

Структура анализатора показана на диаграмме рисунке 15.



Рисунок 15 – Структурная схема анализатора

Представленная структурная схема показывает, как символьный фильтр обрабатывает входной поток строк, удаляя, добавляя или изменяя некоторые символы. Затем обработанные строки передаются на вход токенизатора, который разбивает их на токены (слова). Полученные токены затем проходят через фильтр токенов и подаются на выход.

Для индексов "procedure" и "patient" был использован анализатор, представленный на рисунке 16. 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 16 – Код анализатора для индексации документов

Из рисунка можно заметить, что анализатор содержит токенайзер standard, который удаляет знаки пунктуации, такие как запятые и точки. В данном случае анализатор работает с русским текстом.

Сначала все токены переводятся в нижний регистр с помощью фильтра lowercase. Затем применяются фильтры токенов russian\_stop\_words и snowball.

Фильтр russian\_stop\_words удаляет из строк различные стоп-слова, такие как союзы, предлоги и частицы. Фильтр filter\_ru\_sn типа snowball позволяет выделить основу слов и отбросить окончания, это называется стеммингом (stamming).

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

Ниже приведен пример кода маппинга для индекса procedure:

```
ProcedureMapping = {
  "properties":{
    "name": {
      "type": "text",
      "fielddata": True,
      "analyzer":"analitic_for_ru",
      "search_analyzer":"analitic_for_ru"
    },
    "description": {
      "type": "text",
      "fielddata": True,
      "analyzer":"analitic_for_ru",
      "search_analyzer":"analitic_for_ru"
    },
    "price": {
      "type": "integer"
    }
  }
}
```

Далее приведен код маппинга для индекса patient:

```
PatientMapping = {
  "properties":{
    "patient_id": {
      "type": "text",
      "fielddata": True
    },
    "personal_data": {
      "type": "text",
```

```

        "analyzer":"analitic_for_ru",
        "search_analyzer":"analitic_for_ru",
        "fielddata": True
    },

    "voucher": {
        "type": "text",
        "fielddata": True
    },

    "date_of_arrival": {
        "type": "date",
        "format": "yyyy-MM-dd"
    },

    "number_of_days": {
        "type": "integer"
    },

    "diagnosis":{
        "type": "text",
        "analyzer":"analitic_for_ru",
        "search_analyzer":"analitic_for_ru",
        "fielddata": True
    },

    "precedures_id": {
        "type": "text",
        "fielddata": True
    }
}

```

В таблице 1 показаны типы данных, применяемые в маппингах индексов.



Таблица 1 – Типы данных для индексов

Тип данных	Описание типа данных
date	дата в формате уууу-ММ-dd
integer	целое число
keyword	ключевые слова, структурированные данные к которым не применяется анализатор
text	неструктурированный текст (может быть обработан анализатором)

Описанный процесс индексации документов в Elasticsearch был реализован с помощью Python-скрипта. Алгоритм работы скрипта включает следующие шаги:

- 1) Подключение к кластеру Elasticsearch при помощи библиотеки ES.
- 2) Проверка наличия индексов в кластере. Если индексы не найдены, они создаются. Если же индексы уже существуют, то они удаляются и создаются заново.
- 3) Создание анализатора и указание его настроек.
- 4) Заккрытие 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.

При возникновении исключений процесс индексации прекращается, а на экран выводится информация об ошибке.

Схема работы программы представлена на рисунке 17, а сам код программы можно найти в приложении Б.

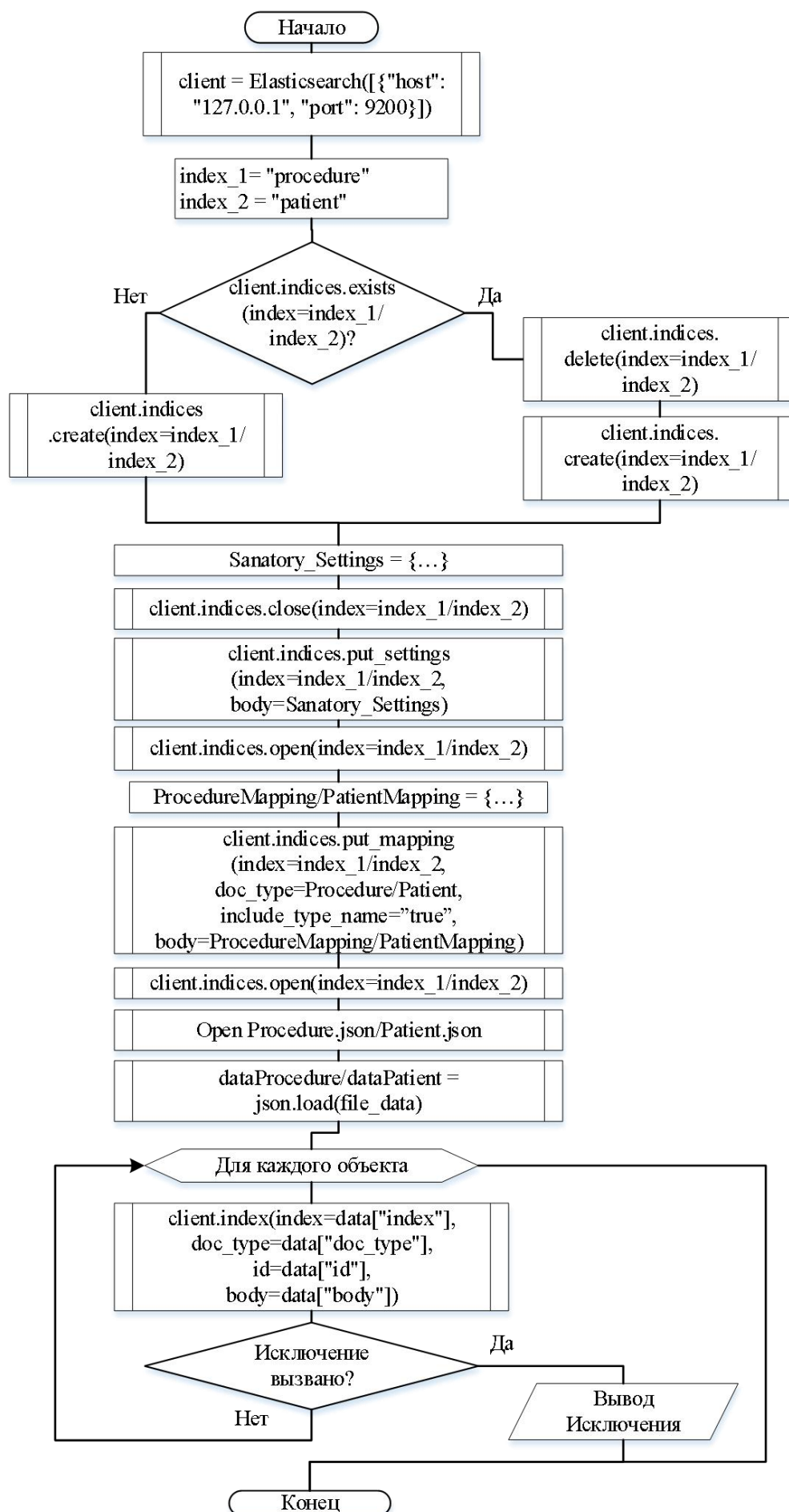


Рисунок 17 – схема алгоритма маппинга и индексации

Результат отработанной программы представлен на рисунке 16.

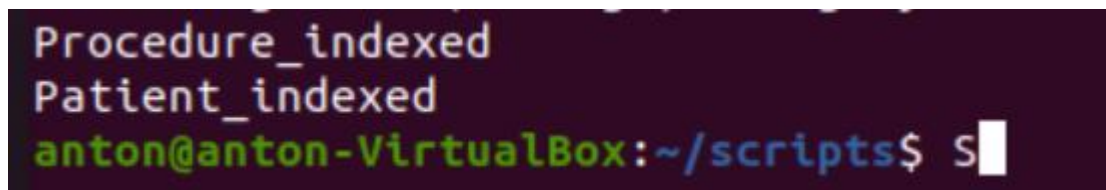


Рисунок 18 – вывод программы маппинга и индексации

Для того чтобы увидеть индексы, их маппинг и анализатор, необходимо перейти по следующим ссылкам: «<http://localhost:9200/procedure>» и «<http://localhost:9200/patient>». Результаты перехода по этим ссылкам можно увидеть на рисунке 19.

JSON	Raw Data	Headers	JSON	Raw Data	Headers
Save	Copy	Collapse All	Save	Copy	Collapse All
Expand All	Filter JSON		Expand All	Filter JSON	
▼ properties:			▼ procedure:		
▼ date_of_arrival:			aliases: {}		
type:	"date"		▼ mappings:		
format:	"yyyy-MM-dd"		▼ properties:		
▼ diagnosis:			▼ description:		
type:	"text"		type:	"text"	
analyzer:	"analitic_for_ru"		analyzer:	"analitic_for_ru"	
fielddata:	true		fielddata:	true	
▼ number_of_days:			▼ name:		
type:	"integer"		type:	"text"	
▼ patient_id:			analyzer:	"analitic_for_ru"	
type:	"text"		fielddata:	true	
fielddata:	true		▼ price:		
▼ personal_data:			type:	"integer"	
type:	"text"		▼ settings:		
analyzer:	"analitic_for_ru"		▼ index:		
fielddata:	true		▼ routing:		
▼ procedures_id:			▼ allocation:		
type:	"text"		▼ include:		
fielddata:	true		_tier_preference:	"data_content"	
▼ voucher:			number_of_shards:	"1"	
type:	"text"		provided_name:	"procedure"	
fielddata:	true		creation_date:	"1681590521892"	
▼ settings:					

Рисунок 19 – Индексы patient и procedure с маппингом и анализатором

### 3.3. Запросы с вложенной агрегацией к данным ES

#### 3.3.1. Первый запрос

Согласно заданию, требуется сделать запрос.

Запрос: разбить пациентов по дате прибытия с периодом 1 год, для каждой «корзины» определить количество пациентов, прошедших каждую процедуру.

Для выполнения запроса была использована следующая реализация:

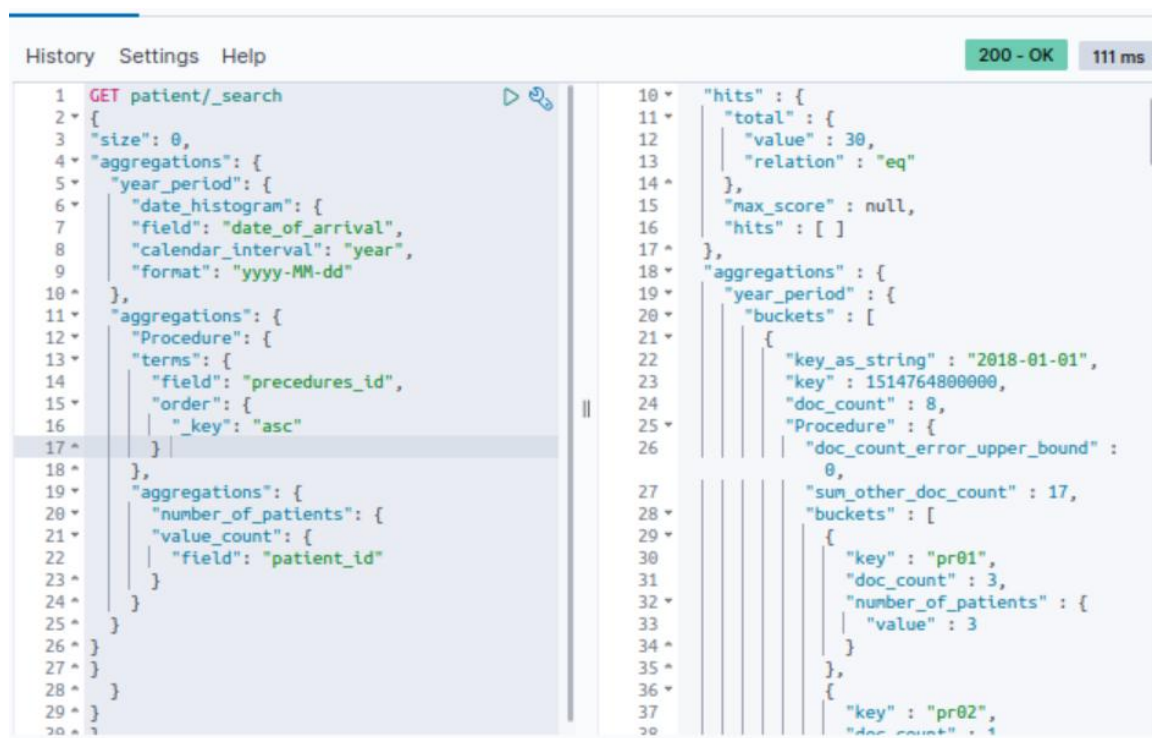
GET patient/\_search

```
{
  "size": 0,
  "aggregations": {
    "year_period": {
      "date_histogram": {
        "field": "date_of_arrival",
        "calendar_interval": "year",
        "format": "yyyy-MM-dd"
      },
    },
    "aggregations": {
      "Procedure": {
        "terms": {
          "field": "precedures_id",
          "order": {
            "_key": "asc"
          }
        },
      },
    },
    "aggregations": {
      "number_of_patients": {
        "value_count": {
          "field": "patient_id"
        },
      },
    },
  },
},
},
},
}
```

Этот запрос использует вложенную агрегацию, где каждый уровень помечен как "aggregations". На первом уровне все документы процедур группируются по годам с помощью временной гистограммы "date\_histogram".

На следующем уровне для каждого года производится группировка по процедуре с использованием поля "precedures\_id". На последнем уровне агрегации определяется стоимость выполненных процедур для каждого специалиста с помощью гистограммы "value\_count" по полю "patient\_id".

Результат выполненного запроса в среде Kibana показан на рисунке 20. Полный вывод запроса доступен в приложении А.



The screenshot displays the Kibana search console interface. The top bar shows 'History', 'Settings', and 'Help' tabs. On the right, a status bar indicates '200 - OK' and '111 ms'. The main area is split into two panels. The left panel contains the query:

```
1 GET patient/_search
2 {
3   "size": 0,
4   "aggregations": {
5     "year_period": {
6       "date_histogram": {
7         "field": "date_of_arrival",
8         "calendar_interval": "year",
9         "format": "yyyy-MM-dd"
10      }
11    },
12    "Procedure": {
13      "terms": {
14        "field": "precedures_id",
15        "order": {
16          "_key": "asc"
17        }
18      },
19      "aggregations": {
20        "number_of_patients": {
21          "value_count": {
22            "field": "patient_id"
23          }
24        }
25      }
26    }
27  }
28 }
29 }
```

The right panel shows the JSON response, which is a partial view of the full result set:

```
10 {
11   "hits": {
12     "total": {
13       "value": 30,
14       "relation": "eq"
15     },
16     "max_score": null,
17     "hits": [ ]
18   },
19   "aggregations": {
20     "year_period": {
21       "buckets": [
22         {
23           "key_as_string": "2018-01-01",
24           "key": "1514764800000",
25           "doc_count": 8,
26           "Procedure": {
27             "doc_count_error_upper_bound": 0,
28             "sum_other_doc_count": 17,
29             "buckets": [
30               {
31                 "key": "pr01",
32                 "doc_count": 3,
33                 "number_of_patients": {
34                   "value": 3
35                 }
36               },
37               {
38                 "key": "pr02",
39                 "doc_count": 1
40               }
41             ]
42           }
43         }
44       ]
45     }
46   }
47 }
```

Рисунок 20 – Результат выполнения первого запроса

### 3.3.2. Второй запрос

Необходимо выполнить следующее задание: определить стоимость процедуры по заданным ключевым словам.

В качестве ключевого слова было выбрано слово "травы". Для поиска данного слова в описании и названии процедур была использована агрегация на основе "simple\_query\_string".

Результат запроса содержит id процедуры, её название и стоимость. Также в запросе отображено найденное слово в описании процедуры и выделено функцией "highlight". Ниже приведен пример реализации запроса.

```
GET procedure/_search
{
  "query": {
    "simple_query_string": {
      "query": "травы"
    }
  },
  "_source": [
    "name",
    "price"
  ],
  "highlight": {
    "fields": {
      "description": {}
    }
  }
}
```

На изображении под номером 21 показан результат выполненного запроса в интерфейсе Kibana. Полный вывод запроса можно найти в приложении А.

History Settings Help

200 - OK 87 ms

```

1 GET procedure/_search
2 {
3   "query": {
4     "simple_query_string": {
5       "query": "травы"
6     }
7   },
8   "_source": [
9     "name",
10    "price"
11  ],
12  "highlight": {
13    "fields": {
14      "description": {}
15    }
16  }

```

```

9  },
10  "hits": {
11    "total": {
12      "value": 3,
13      "relation": "eq"
14    },
15    "max_score": 5.1222467,
16    "hits": [
17      {
18        "_index": "procedure",
19        "_type": "Procedure",
20        "_id": "PR18",
21        "_score": 5.1222467,
22        "_source": {
23          "price": "7000",
24          "name": "Терапия травами"
25        },
26        "highlight": {
27          "description": [
28            "Процедура, основанная на
использовании лечебных <em>трав
</em> и растений для оздоровления
организма, укрепления"
29          ]
30        }
31      },
32      { },
33      { }
47    ]
62  }

```

Рисунок 21 – Результат выполнения второго запроса

### 3.4 Neo4j

#### 3.4.1 Создание и заполнение графовой базы данных

Перед заполнением графовой БД Neo4j необходимо получить данные из Elasticsearch. После этого были созданы узлы.

В итоговой графовой БД были созданы следующие сущности:

- Узел: «Patient\_Node» (Пациент): «Patient\_id» (id пациента), «Patient\_personal\_data» (персональные данные пациента), «Voucher» (номер путевки), «Arrival\_date» (дата прибытия), «Numb\_of\_days» (количество дней, проведенных в санатории), «Diagnosis» (диагнозы пациента) и «Procedures\_id» (id процедур, пройденных клиентом);
- Узел: «Procedure\_Node» (Процедура): «Procedure\_id» (id\_процедуры), «Name» (название процедуры), «Description» (описание процедуры), «Price» (цена процедуры).
- Связь: «Got\_procedure» (Прошёл): «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 "Пациент".
- 5) Во вложенном цикле проверяется наличие процедуры в базе данных, и создается новая процедура при ее отсутствии.
- 6) Создается новая связь, если идентификатор процедуры во внутреннем цикле совпадает с одним из идентификаторов Procedures\_id во внешнем цикле.

Схема алгоритма программы для заполнения базы данных Neo4j представлена на рисунке 22. Код программы можно найти в приложении Б.



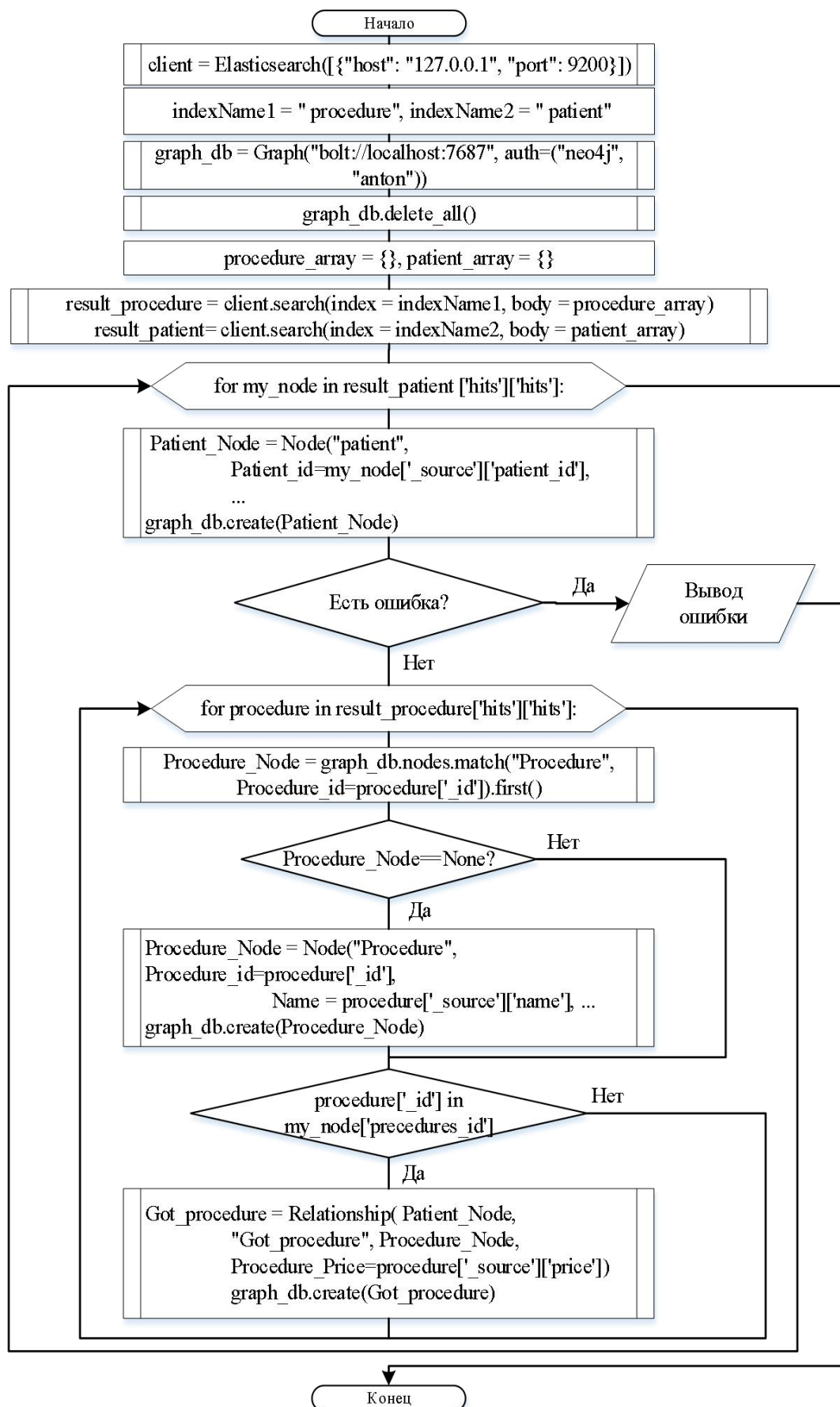


Рисунок 22 – Схема алгоритма создания графовой БД Neo4j

Для просмотра БД, которая была получена, необходимо перейти по адресу <http://localhost:7474/browser>. Чтобы найти связь Пациент-Прошел-Процедуру, которую нужно было создать согласно заданию, следует выполнить запрос на языке Cypher: `MATCH p=()-[r:connect_Got_procedure]->() RETURN p LIMIT 80`. Результат запроса представлен на рисунке 23. На данном рисунке можно заметить, что узлы типа Процедура имеют оранжевый цвет, а узлы типа Пациент - фиолетовый.

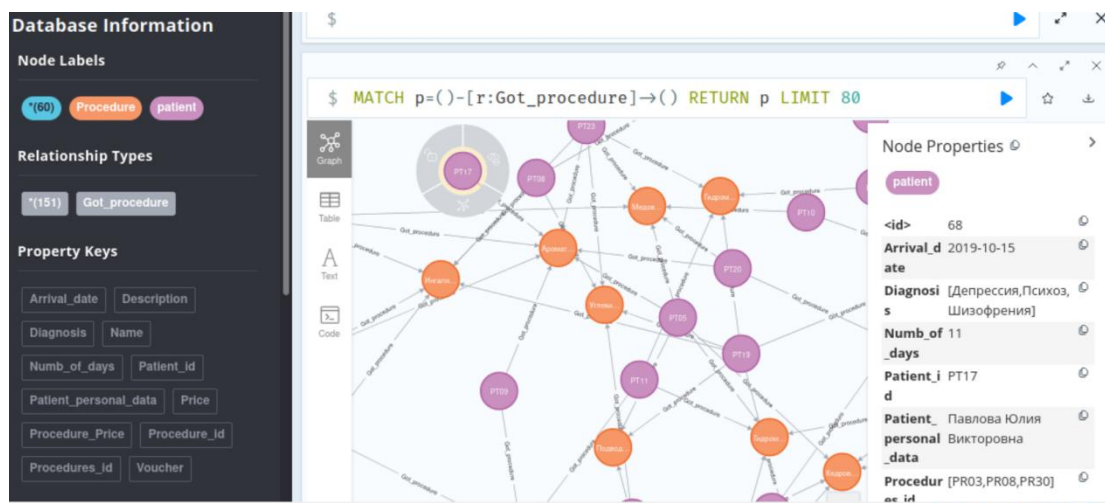


Рисунок 23 – Граф связи Пациент-Прошел-Процедуру БД Neo4j

Связи узлов обозначены серыми стрелками. На рисунке 24 выделена одна из связей. Справа отображены поля, хранящиеся в данной связи, где можно увидеть цену процедуры.

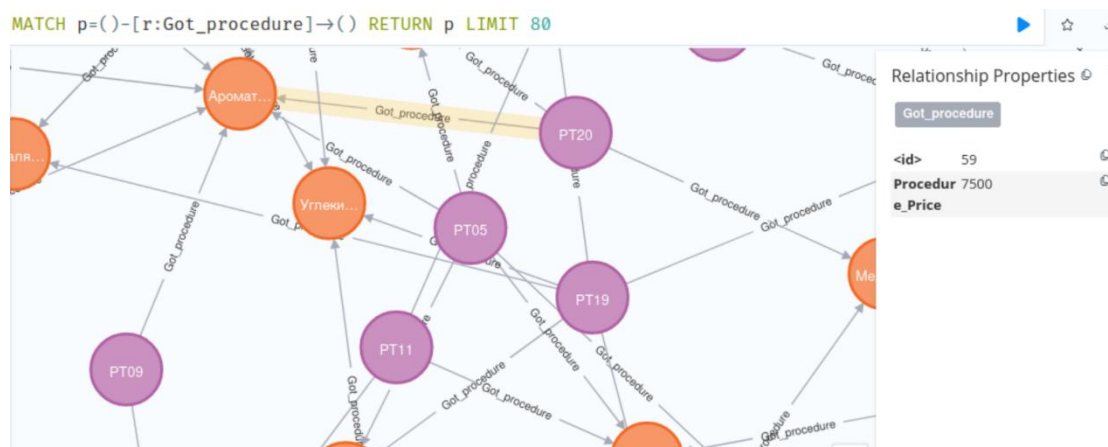


Рисунок 24 – Связь Пациент-Прошел-Процедуру

### 3.4.2 Запрос к графовой базе данных

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

Запрос на языке Cypher:

```
MATCH p=(pct:patient)-[r:Got_procedure]->(proc:Procedure)
WITH proc, sum(toInteger(r.Procedure_Price)) as procedure_sum, count(r) as
number_of_client
ORDER BY procedure_sum desc
RETURN proc.Name, procedure_sum, proc.Price, number_of_client,
proc.Description
LIMIT 1
```

Этот запрос использует связи типа "Пациент-Прошел-Процедуру". Затем для каждой процедуры (proc) рассчитывается суммарная стоимость, указанная в каждой связи. Результаты выводятся в порядке убывания (desc). Ограничивая результат одной записью, мы получаем процедуру с максимальной суммарной стоимостью.

На рисунке 25 показан результат выполнения этого запроса.



The screenshot displays the Cypher query and its result in the Neo4j interface. The query is as follows:

```
1 MATCH p=(pct:patient)-[r:Got_procedure]->(proc:Procedure)
2 WITH proc, sum(toInteger(r.Procedure_Price)) as procedure_sum, count(r) as
  number_of_client
3 ORDER BY procedure_sum desc
4 RETURN proc.Name, procedure_sum, proc.Price, number_of_client, proc
5 LIMIT 1
```

The result table shows the following data:

proc.Name	procedure_sum	proc.Price	number_of_client	proc
"Термальные ванны"	80000	"8000"	10	{ "identity": 1, "labels": [ "Procedure" ], "properties": { "Description": "Процедура, основанная на принятии термальных ванн с использованием теплой или горячей воды с добавлением различных минералов и микроэлементов, что способствует расслаблению"  } }

Рисунок 25 – Результат выполнения запроса к БД Neo4j

## 4. Spark

### 4.1.Создание и заполнение таблиц

В соответствии с заданием требуется создать три CSV-файла с определенными схемами таблиц «Пациент», «Назначение», «Процедура» и сохранить их в файловой системе HDFS. Для этого была разработана программа с следующим алгоритмом:

- 1) Установить соединение с кластером Elasticsearch при помощи API;
- 2) Подключиться к сессии Spark;
- 3) Получить все документы из Elasticsearch при помощи запросов;
- 4) Создать схему данных, которая соответствует каждой таблице и задать типы полей;
- 5) Заполнить таблицы с помощью циклов и преобразования полей документов в соответствии с созданными ранее схемами;
- 6) Создать датафреймы на основе схем и данных, полученных на предыдущем шаге;
- 7) Записать датафреймы в CSV-файлы HDFS, задав параметры, включая заголовки столбцов и разрешение на перезапись.

На рисунке 26 представлена схема алгоритма программы, написанной для создания CSV-файлов, а код этой программы приведен в приложении Б.

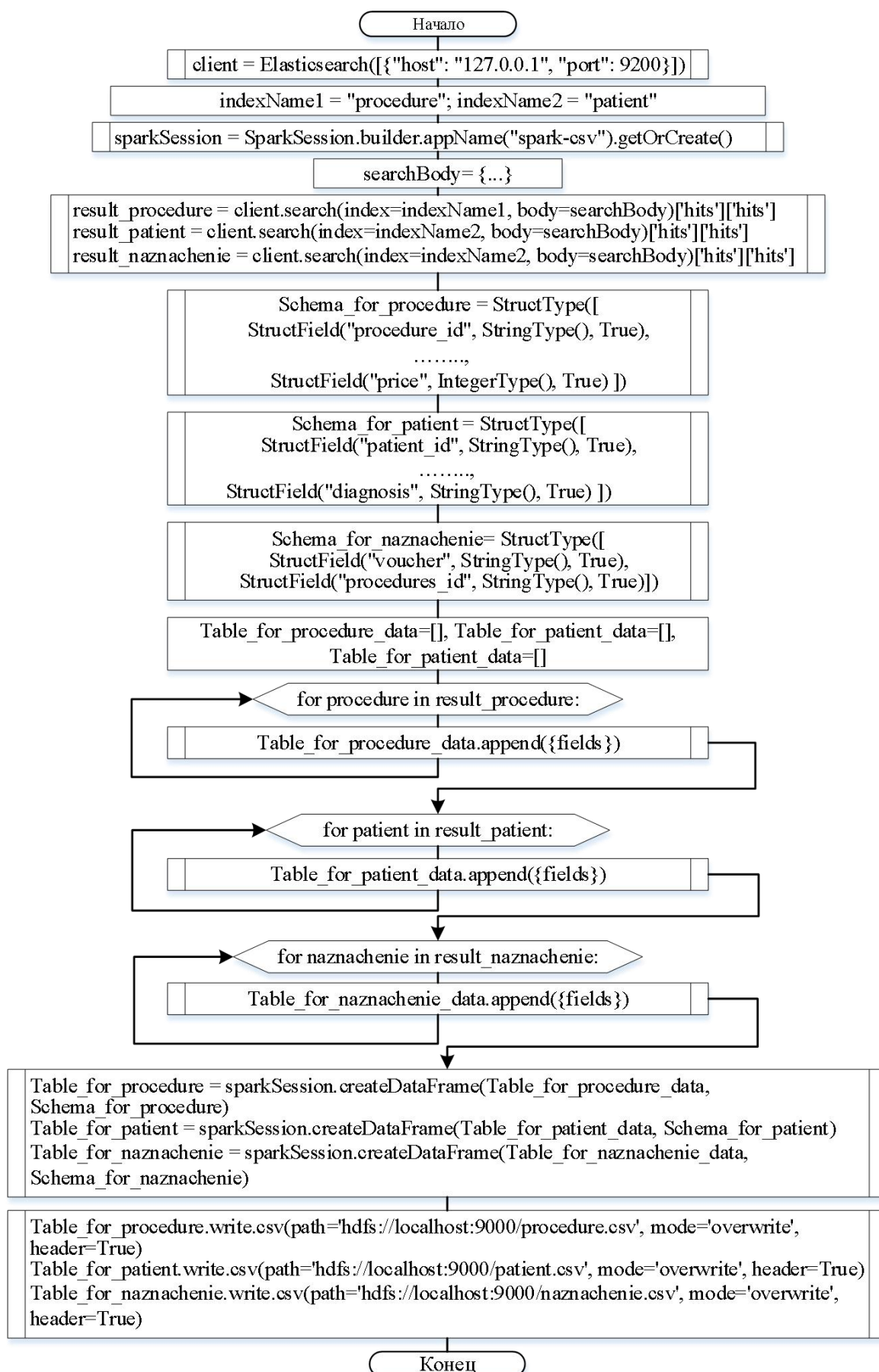


Рисунок 27 – Схема алгоритма программы создания csv-файлов



После выполнения программы были получены таблицы, созданные в соответствии с заданием. Вывод таблиц представлен на рисунках 28, 29 и 30.

```
at org.apache.hadoop.hdfs.DataStreamer.run(DataStreamer.java:810)
Stage 3:> (0 + 1)
```

procedure_id	name	description	price
PR01	Обертывание в вод...	Процедура, основа...	7500
PR02	Термальные ванны	Процедура, основа...	8000
PR03	Ароматерапия	Процедура, основа...	7500
PR04	Грязелечение	Процедура, основа...	9000
PR05	Подводный душ-массаж	Процедура, основа...	6200
PR06	Кедровая бочка	Процедура, основа...	15000
PR07	Углекислые ванны	Процедура, основа...	8000
PR08	Лимфодренажный ма...	Процедура, основа...	5500
PR09	Грязевое обертывание	Процедура, основа...	7000
PR10	Талассотерапия	Процедура, основа...	12000
PR11	Массаж спины и ше...	Процедура, основа...	5000
PR12	Ингаляции с медик...	Процедура, основа...	4500
PR13	Гидромассаж	Процедура, основа...	6500
PR14	Фитобочка	Процедура, основа...	8500
PR15	Антицеллюлитный м...	Процедура, основа...	7500
PR16	Грязелечение	Процедура, основа...	8000
PR17	Углекислые ванны	Процедура, основа...	5500
PR18	Терапия травами	Процедура, основа...	7000
PR19	Кедровая бочка	Процедура, основа...	9000
PR20	Медовый массаж	Процедура, основа...	6000

only showing top 20 rows

Рисунок 28 – Вывод файла procedure.csv

patient_id	personal_data	voucher	diagnosis
PT01	Шаповалова Мария ...	1018765	[Гипертоническая ...
PT02	Иванов Иван Иванович	2309851	[Остеохондроз поз...
PT03	Петров Петр Петрович	9876512	[Бронхиальная аст...
PT04	Сидорова Екатерина...	5123498	[Гинекологические...
PT05	Лебедев Андрей Ни...	7654321	[Ожирение, Сахарн...
PT06	Новикова Ольга Ви...	8765432	[Остеоартрит, Ост...
PT07	Кузнецов Дмитрий ...	6543210	[Гастрит, Язва же...
PT08	Михайлова Елена С...	9876543	[Почечная недоста...
PT09	Смирнов Алексей В...	4321567	[Депрессия, Панич...
PT10	Иванов Андрей Пет...	7654321	[Бронхит, Астма, ...
PT11	Сергеева Екатерина...	8765432	[Онкологическое з...
PT12	Петров Сергей Ива...	9876543	[Ишемическая боле...
PT13	Макарова Ольга Ст...	5432109	[Остеохондроз, Ск...
PT14	Иванова Наталья С...	9870123	[Диабет, Ожирение...
PT15	Петрова Олеся Мих...	5678901	[Остеопороз, Ревм...
PT16	Сидоров Павел Ана...	2345678	[Хронический гаст...
PT17	Павлова Юлия Викт...	4567890	[Депрессия, Психо...
PT18	Григорьев Владими...	7654321	[Простуда, Грипп, ...
PT19	Михайлов Алексей ...	9876543	[Острый аппендици...
PT20	Николаева Елена В...	5432109	[Бронхит, Астма, ...

only showing top 20 rows

Рисунок 29 – Вывод файла patient.csv

```

+-----+-----+
|voucher|      procedures_id|
+-----+-----+
|1018765|[PR01, PR07, PR12]|
|2309851|[PR02, PR08, PR15...|
|9876512|[PR04, PR10, PR18...|
|5123498|[PR06, PR09, PR11...|
|7654321|[PR03, PR05, PR13...|
|8765432|[PR07, PR09, PR14...|
|6543210|[PR02, PR08, PR16...|
|9876543|[PR04, PR12, PR17...|
|4321567|[PR01, PR03, PR30]|
|7654321|[PR02, PR06, PR15...|
|8765432|[PR01, PR07, PR13...|
|9876543|[PR02, PR08, PR14...|
|5432109|[PR01, PR05, PR11...|
|9870123|[PR02, PR04, PR09...|
|5678901|[PR11, PR19, PR27...|
|2345678|[PR02, PR07, PR13...|
|4567890|[PR03, PR08, PR30]|
|7654321|[PR01, PR06, PR18...|
|9876543|[PR02, PR05, PR12...|
|5432109|[PR03, PR08, PR15...|
+-----+-----+
only showing top 20 rows

```

Рисунок 30 – Вывод файла naznachenie.csv

## 4.2. Запрос в Spark

По заданию требуется реализовать запрос SELECT: определить суммарное число назначений по каждой процедуре. На данный момент в Spark присутствуют следующие таблицы «procedure», «patient» и «naznachenie».

Используя данные таблицы построим запрос:

```

SELECT ID_OF_PROCEDURE, COUNT(ID_OF_PROCEDURE) as NUMB_OF_NAZN
FROM (

```



```

SELECT procedure.procedure_id as ID_OF_PROCEDURE,
naznachenie.procedures_id LIKE concat(concat('%',
procedure.procedure_id), '%') as CHECK
FROM procedure, naznachenie)
WHERE check=true
GROUP BY ID_OF_PROCEDURE
ORDER BY ID_OF_PROCEDURE

```

Для выполнения запроса была разработана программа на Python, следующим образом:

- 1) Установка соединения с сессией Spark;
- 2) Чтение файла procedure.csv;
- 3) Создание временной таблицы;
- 4) Выполнение SQL-запроса.

Код программы для выполнения запроса приведен в приложении Б. На рисунке 31 отображается результат запроса.

ID_OF_PROCEDURE	NUMBER_OF_NAZNACH
PR01	9
PR02	10
PR03	7
PR04	3
PR05	3
PR06	5
PR07	7
PR08	9
PR09	4
PR10	1
PR11	6
PR12	6
PR13	4
PR14	5
PR15	3
PR16	1
PR17	4
PR18	7
PR19	5
PR20	4
PR22	4
PR23	5
PR24	5

Рисунок 31 – Результат выполнения запроса в Spark



### 4.3. Мониторинг Spark

Для отслеживания состояния и производительности задач в Spark был осуществлен переход на веб-интерфейс по адресу <http://127.0.0.1:4040/jobs/>.

Когда программа с запросом запускается, Spark предоставляет информацию о временных характеристиках выполнения задач данного запроса. На рисунке 32 представлена временная диаграмма пятого job-а, который относится к данному запросу.



Рисунок 32 – Временная диаграмма выполнения запроса

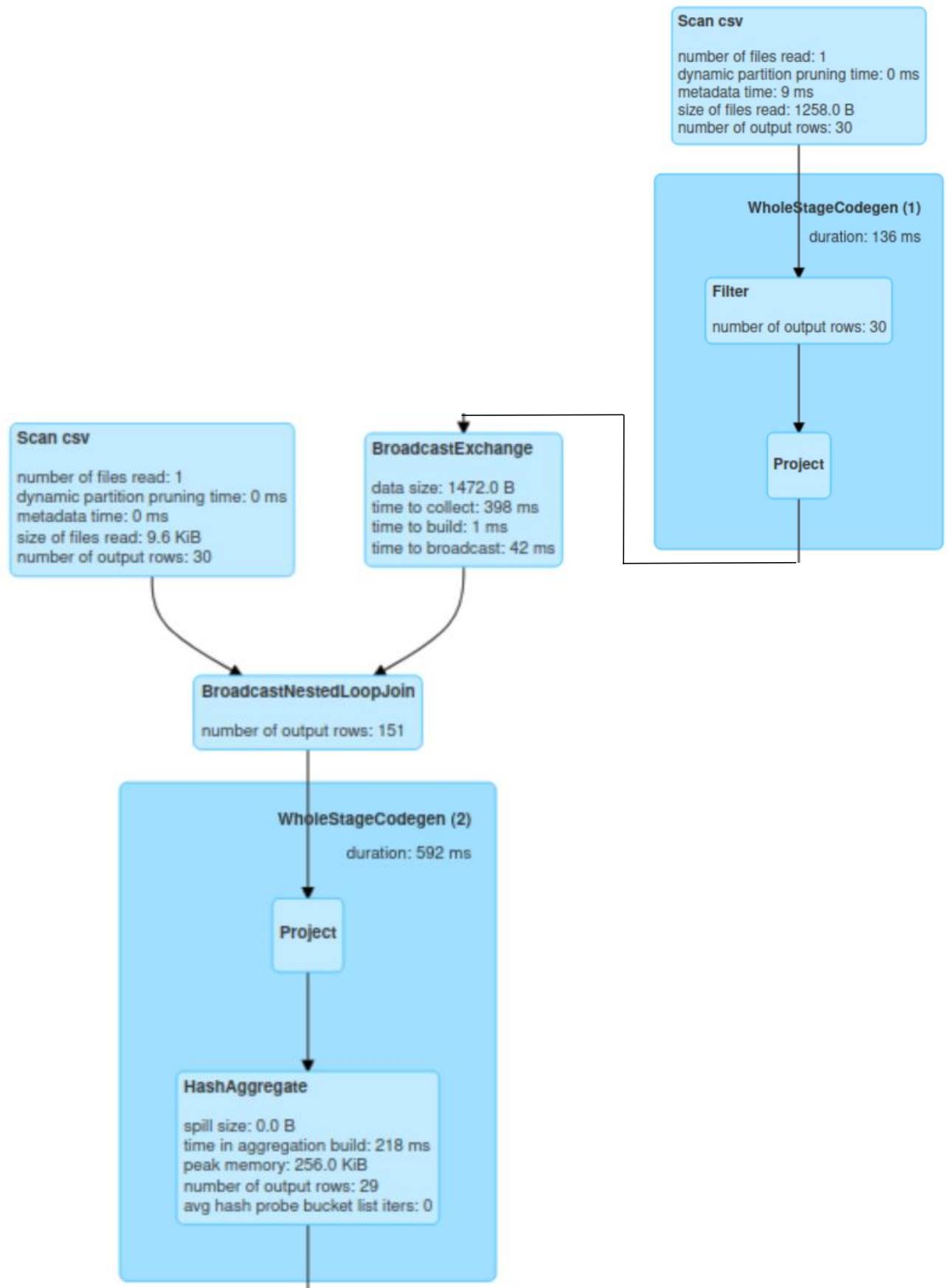
Из временной диаграммы видно, что время выполнения запроса составило 0,7 секунды (Total Tasks Time), причем количество задач, которые были выполнены, зависело от количества разделов с записями (partitions). Чтобы уменьшить время выполнения запроса, было изменено число разделов с записями на 100, что позволило сократить время выполнения запроса. В разделе SQL представлены данные о всех выполненных запросах, которые можно увидеть на рисунке 33.

	Description		Submitted	Duration	Job IDs
	showString at NativeMethodAccessorImpl.java:0	+details	2023/05/05 17:55:40	5 s	[4][5]
3	createOrReplaceTempView at NativeMethodAccessorImpl.java:0	+details	2023/05/05 17:55:39	5 ms	
2	createOrReplaceTempView at NativeMethodAccessorImpl.java:0	+details	2023/05/05 17:55:39	5 ms	
1	load at NativeMethodAccessorImpl.java:0	+details	2023/05/05 17:55:39	0.1 s	[2]
0	load at NativeMethodAccessorImpl.java:0	+details	2023/05/05 17:55:36	2 s	[0]

Рисунок 33 – SQL-запросы сессии в Spark

Основным запросом задания является тот, что имеет ID=4. Разберем его более подробно. Стадии выполнения данного запроса и его временные

характеристики представлены в виде DAG (Ориентированного ациклического графа). На рисунке 34 представлен DAG основного SQL-запроса. Физический план выполнения запроса представлен в приложении Б.



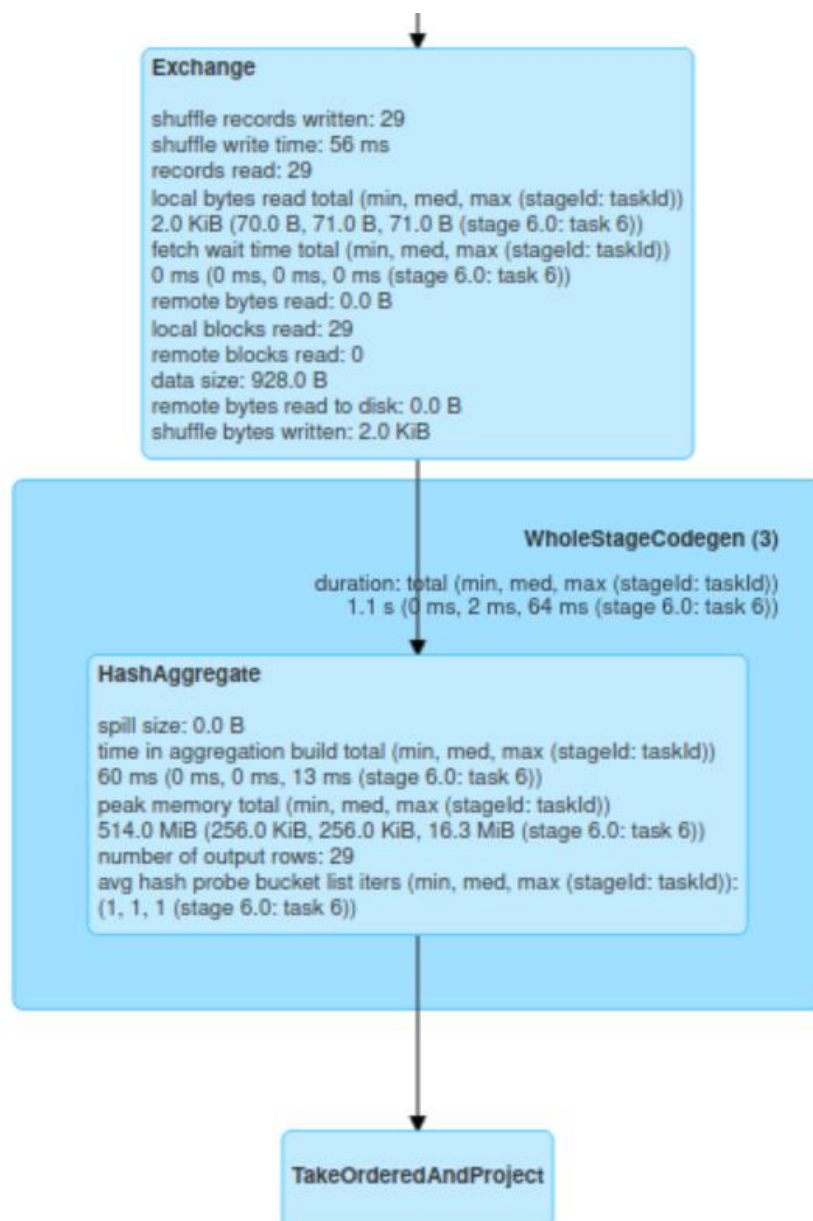


Рисунок 34 – DAG SQL-запроса

Выполнение SQL-запроса начинается с чтения таблиц «процедура» и «назначение» из файлов procedure.csv и naznachenie.csv. Было просканировано по 30 строк каждой таблицы. Далее выполняется BroadcastNestedLoopJoin. Эта функция формирует новую таблицу с двумя столбцами (id процедуры и check). Check хранит булевы значения - true, false для каждой процедуры относительно каждого назначения.

Далее выполняется HashAggregate. Эта функция оставляет строки со значением true. Полученный результат поступает в Exchange, где происходит слияние данных из разных процессов и разделов. Потом в разделе WholeStageCodegen подраздел HashAggregate выполняет группировку записей.

## ЗАКЛЮЧЕНИЕ

В результате работы над курсовым проектом был создан макет аналитической системы санатория, который основан на базах данных Elasticsearch и Neo4j, а также на сервисах Hadoop и Spark. Для работы были установлены необходимые программные средства на VM Virtual Box под управлением ОС Ubuntu-20.

В ходе работы были сгенерированы документы типа "Процедура" и "Пациент", а также развернут кластер Elasticsearch с двумя индексами - Procedure и Patient. Для проиндексированных документов была реализована программа маппинга с анализатором русского текста. С использованием среды Kibana были сформированы и выполнены запросы с вложенной агрегацией.

Кроме того, была создана и заполнена графовая база данных Neo4j, в которой были сформированы узлы "Процедура" и "Пациент", а также связи между ними. Связь "Пациент-Получил-Процедуру", требуемая по заданию, была продемонстрирована в графическом виде, а также был реализован запрос на языке Cypher.

Для работы с Spark в файловой системе HDFS были созданы CSV-файлы, содержащие таблицы "Пациент", "Процедура" и "Назначение". Был реализован SQL-запрос, который был проанализирован при помощи монитора Spark. В итоге был разработан макет аналитической системы санатория, основанный на базах данных NoSQL.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Neo4j – Графовая база данных. – URL: <https://neo4j.com/what-is-a-graph-database> (дата обращения 14.03.2023)
2. Hadoop: официальный сайт. – URL: <https://hadoop.apache.org/> (дата обращения 13.03.2023)
3. Apache Spark. – URL: <https://www.bigdataschool.ru/wiki/spark> (дата обращения 17.03.2023)
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 20.03.2023)
6. Сайт Генератора ФИО - URL: <https://randomus.ru> (дата обращения 17.03.2022)
7. Онлайн-генератором JSON-документов URL: <https://app.json-generator.com/> (дата обращения 16.03.2023)
8. Официальная документация Elasticsearch – URL: <https://www.elastic.co/guide/en/elasticsearch/reference/current/getting-started-index.html> (дата обращения 17.04.2023).

## **ПРИЛОЖЕНИЕ А**

Созданные JSON-файлы и результаты выполнения запросов

## 1 Созданные JSON документы типа «Процедура»

```
[
{
  "index": "procedure",
  "doc_type": "Procedure",
  "id": "PR01",
  "body": {
    "name": "Обертывание в водоросли",
    "description": "Процедура, основанная на обертывании тела в водоросли с высоким содержанием микроэлементов и биологически активных веществ, для питания, увлажнения, улучшения тонуса кожи и омоложения организма.",
    "price": "7500"
  }
},
{
  "index": "procedure",
  "doc_type": "Procedure",
  "id": "PR02",
  "body": {
    "name": "Термальные ванны",
    "description": "Процедура, основанная на принятии термальных ванн с использованием теплой или горячей воды с добавлением различных минералов и микроэлементов, что способствует расслаблению мышц, улучшению кровообращения и общего состояния.",
    "price": "8000"
  }
},
{
  "index": "procedure",
  "doc_type": "Procedure",
  "id": "PR03",
  "body": {
    "name": "Ароматерапия",
    "description": "Процедура, основанная на использовании эфирных масел различных ароматов для расслабления и оздоровления организма.",
    "price": "7500"
  }
},
{
  "index": "procedure",
  "doc_type": "Procedure",
  "id": "PR04",
  "body": {
    "name": "Грязелечение",
    "description": "Процедура, основанная на использовании приложения грязи на кожу, что способствует улучшению состояния кожи и организма в целом.",
    "price": "9000"
  }
},
{
  "index": "procedure",
  "doc_type": "Procedure",
  "id": "PR05",
  "body": {
    "name": "Подводный душ-массаж",
    "description": "Процедура, основанная на воздействии водного массажа на тело, что способствует расслаблению мышц и улучшению кровообращения.",
    "price": "6200"
  }
},
{
  "index": "procedure",
  "doc_type": "Procedure",
  "id": "PR06",
  "body": {
    "name": "Кедровая бочка",
    "description": "Процедура, основанная на воздействии на организм микроэлементов, содержащихся в кедровой древесине, что способствует укреплению иммунной системы и омоложению организма.",
    "price": "15000"
  }
},
{
  "index": "procedure",
  "doc_type": "Procedure",
  "id": "PR07",
  "body": {
    "name": "Углекислые ванны",
    "description": "Процедура, основанная на использовании углекислых ванн для улучшения состояния кожи, активации обменных процессов и улучшения общего самочувствия.",
    "price": "8000"
  }
},
{
  "index": "procedure",
```

```

"doc_type": "Procedure",
"id": "PR08",
"body": {
"name": "Лимфодренажный массаж",
"description": "Процедура, основанная на мягком воздействии на лимфатическую систему, что способствует выведению токсинов и улучшению общего здоровья.",
"price": "5500"
}, {
"index": "procedure",
"doc_type": "Procedure",
"id": "PR09",
"body": {
"name": "Грязевое обертывание",
"description": "Процедура, основанная на обертывании тела грязью, что способствует омоложению кожи, улучшению ее тонуса и увлажнению.",
"price": "7000"
}, {
"index": "procedure",
"doc_type": "Procedure",
"id": "PR10",
"body": {
"name": "Талассотерапия",
"description": "Процедура, основанная на использовании морской воды, морского воздуха и морских продуктов для оздоровления организма и повышения иммунитета.",
"price": "12000"
}, {
"index": "procedure",
"doc_type": "Procedure",
"id": "PR11",
"body": {
"name": "Массаж спины и шейно-воротниковой зоны",
"description": "Процедура, основанная на массаже спины и шеи, что способствует расслаблению мышц, улучшению кровообращения и снятию напряжения.",
"price": "5000"
}, {
"index": "procedure",
"doc_type": "Procedure",
"id": "PR12",
"body": {
"name": "Ингаляции с медикаментами",
"description": "Процедура, основанная на вдыхании медикаментов для лечения заболеваний дыхательной системы, таких как бронхит и астма.",
"price": "4500"
}, {
"index": "procedure",
"doc_type": "Procedure",
"id": "PR13",
"body": {
"name": "Гидромассаж",
"description": "Процедура, основанная на массаже тела водой под высоким давлением, что способствует расслаблению мышц и улучшению кровообращения.",
"price": "6500"
}, {
"index": "procedure",
"doc_type": "Procedure",
"id": "PR14",
"body": {
"name": "Фитобочка",
"description": "Процедура, основанная на воздействии на организм паром с добавлением лечебных трав, что способствует очищению кожи, укреплению иммунной системы и омоложению организма.",
"price": "8500"
}, {
"index": "procedure",
"doc_type": "Procedure",
"id": "PR15",
"body": {
"name": "Антицеллюлитный массаж",
"description": "Процедура, основанная на специальных техниках массажа, направленных на борьбу с целлюлитом и улучшение состояния кожи, способствует улучшению метаболизма и снижению объемов.",
"price": "7500"
}, {
"index": "procedure",

```



```

"doc_type": "Procedure",
"id": "PR16",
"body": {
"name": "Грязелечение",
"description": "Процедура, основанная на нанесении грязевых аппликаций на кожу, что способствует улучшению состояния кожи, активизации обменных процессов и омоложению организма.",
"price": "8000"
}, {
"index": "procedure",
"doc_type": "Procedure",
"id": "PR17",
"body": {
"name": "Углекислые ванны",
"description": "Процедура, основанная на насыщении организма углекислым газом через ванны, что способствует расслаблению мышц, улучшению кровообращения и омоложению организма.",
"price": "5500"
}, {
"index": "procedure",
"doc_type": "Procedure",
"id": "PR18",
"body": {
"name": "Терапия травами",
"description": "Процедура, основанная на использовании лечебных трав и растений для оздоровления организма, укрепления иммунной системы и снятия стресса.",
"price": "7000"
}, {
"index": "procedure",
"doc_type": "Procedure",
"id": "PR19",
"body": {
"name": "Кедровая бочка",
"description": "Процедура, основанная на пребывании в специальной бочке из кедрового дерева, что способствует расслаблению, улучшению состояния кожи и повышению иммунитета.",
"price": "9000"
}, {
"index": "procedure",
"doc_type": "Procedure",
"id": "PR20",
"body": {
"name": "Медовый массаж",
"description": "Процедура, основанная на массаже тела медом, что способствует улучшению состояния кожи, активизации обменных процессов и омоложению организма.",
"price": "6000"
}, {
"index": "procedure",
"doc_type": "Procedure",
"id": "PR21",
"body": {
"name": "Ароматерапия",
"description": "Процедура, основанная на использовании эфирных масел для создания ароматической среды, которая способствует расслаблению, снятию стресса и улучшению настроения.",
"price": "5000"
}, {
"index": "procedure",
"doc_type": "Procedure",
"id": "PR22",
"body": {
"name": "Гидромассаж",
"description": "Процедура, основанная на массаже тела с использованием водных струй, что способствует расслаблению мышц, улучшению кровообращения и омоложению организма.",
"price": "7000"
}, {
"index": "procedure",
"doc_type": "Procedure",
"id": "PR23",
"body": {
"name": "Иглорефлексотерапия",
"description": "Процедура, основанная на воздействии на определенные точки на теле иглами, что способствует снятию болевых синдромов, улучшению общего состояния и повышению иммунитета.",
"price": "8000"
}, {
"index": "procedure",

```

```

"doc_type": "Procedure",
"id": "PR24",
"body": {
"name": "Парафинотерапия",
"description": "Процедура, основанная на нанесении теплого парафинового покрытия на кожу, что способствует увлажнению, питанию и омоложению кожи.",
"price": "4500"
}, {
"index": "procedure",
"doc_type": "Procedure",
"id": "PR25",
"body": {
"name": "Фитотерапия",
"description": "Процедура, основанная на использовании лечебных растений и трав для оздоровления организма, укрепления иммунной системы и повышения энергетического потенциала.",
"price": "6500"
}, {
"index": "procedure",
"doc_type": "Procedure",
"id": "PR26",
"body": {
"name": "Озонотерапия",
"description": "Процедура, основанная на введении озона в организм, что способствует улучшению обменных процессов, активизации иммунной системы и омоложению организма.",
"price": "9000"
}, {
"index": "procedure",
"doc_type": "Procedure",
"id": "PR27",
"body": {
"name": "Талассотерапия",
"description": "Процедура, основанная на использовании морской воды, морских водорослей и других морских компонентов для оздоровления организма, улучшения состояния кожи, снятия стресса и усталости.",
"price": "8500"
}, {
"index": "procedure",
"doc_type": "Procedure",
"id": "PR28",
"body": {
"name": "Медовый массаж",
"description": "Процедура, основанная на массаже тела с использованием меда, что способствует улучшению кровообращения, питанию кожи и омоложению тела.",
"price": "7000"
}, {
"index": "procedure",
"doc_type": "Procedure",
"id": "PR29",
"body": {
"name": "Криотерапия",
"description": "Процедура, основанная на воздействии на организм экстремально низкими температурами, что способствует укреплению иммунной системы, улучшению состояния кожи и общего состояния организма.",
"price": "7500"
}, {
"index": "procedure",
"doc_type": "Procedure",
"id": "PR30",
"body": {
"name": "Массаж спины и шейно-воротниковой зоны",
"description": "Процедура, основанная на массаже спины и шейно-воротниковой зоны, что способствует расслаблению мышц, снятию болевых ощущений и улучшению общего состояния.",
"price": "5500"
}
}]

```

## 2 Созданные JSON документы типа «Пациент»

```
[
  {
    "index": "patient",
    "doc_type": "Patient",
    "id": "PT01",
    "body": {
      "patient_id": "PT01",
      "personal_data": "Шаповалова Мария Александровна",
      "voucher": "1018765",
      "date_of_arrival": "2018-01-22",
      "number_of_days": "10",
      "diagnosis": [
        "Гипертоническая болезнь",
        "Мигрень"
      ],
      "precedures_id": [
        "PR01",
        "PR07",
        "PR12"
      ]
    }
  },
  {
    "index": "patient",
    "doc_type": "Patient",
    "id": "PT02",
    "body": {
      "patient_id": "PT02",
      "personal_data": "Иванов Иван Иванович",
      "voucher": "2309851",
      "date_of_arrival": "2019-06-12",
      "number_of_days": "15",
      "diagnosis": [
        "Остеохондроз позвоночника",
        "Расстройства опорно-двигательного аппарата",
        "Неврозы и депрессии"
      ],
      "precedures_id": [
        "PR02",
        "PR08",
        "PR15",
        "PR27"
      ]
    }
  },
  {
    "index": "patient",
    "doc_type": "Patient",
    "id": "PT03",
    "body": {
      "patient_id": "PT03",
      "personal_data": "Петров Петр Петрович",
      "voucher": "9876512",
      "date_of_arrival": "2020-09-28",
      "number_of_days": "7",
      "diagnosis": [
        "Бронхиальная астма",
        "Хронический бронхит",
        "Хронический панкреатит"
      ],
      "precedures_id": [
        "PR04",
        "PR10",
        "PR18",
        "PR23"
      ]
    }
  },
  {
    "index": "patient",
    "doc_type": "Patient",
    "id": "PT04",
    "body": {
      "patient_id": "PT04",
      "personal_data": "Сидорова Екатерина Васильевна",
      "voucher": "5123498",
```

```

      "date_of_arrival": "2021-03-15",
      "number_of_days": "20",
      "diagnosis": [
        "Гинекологические заболевания",
        "Бесплодие",
        "Хроническая сердечная недостаточность",
        "Расстройства сна"
      ],
      "precedures_id": [
        "PR06",
        "PR09",
        "PR11",
        "PR20",
        "PR26"
      ]
    }
  },
  {
    "index": "patient",
    "doc_type": "Patient",
    "id": "PT05",
    "body": {
      "patient_id": "PT05",
      "personal_data": "Лебедев Андрей Николаевич",
      "voucher": "7654321",
      "date_of_arrival": "2018-09-03",
      "number_of_days": "14",
      "diagnosis": [
        "Ожирение",
        "Сахарный диабет",
        "Гипертоническая болезнь"
      ],
      "precedures_id": [
        "PR03",
        "PR05",
        "PR13",
        "PR19",
        "PR28"
      ]
    }
  },
  {
    "index": "patient",
    "doc_type": "Patient",
    "id": "PT06",
    "body": {
      "patient_id": "PT06",
      "personal_data": "Новикова Ольга Викторовна",
      "voucher": "8765432",
      "date_of_arrival": "2019-11-25",
      "number_of_days": "12",
      "diagnosis": [
        "Остеоартрит",
        "Остеопороз",
        "Ревматоидный артрит"
      ],
      "precedures_id": [
        "PR07",
        "PR09",
        "PR14",
        "PR24",
        "PR29"
      ]
    }
  },
  {
    "index": "patient",
    "doc_type": "Patient",
    "id": "PT07",
    "body": {
      "patient_id": "PT07",
      "personal_data": "Кузнецов Дмитрий Алексеевич",
      "voucher": "6543210",
      "date_of_arrival": "2020-06-08",
      "number_of_days": "8",
      "diagnosis": [
        "Гастрит",
        "Язва желудка",
        "Хронический панкреатит",
        "Холецистит"
      ]
    }
  }
}

```

```

    ],
    "precedures_id": [
        "PR02",
        "PR08",
        "PR16",
        "PR22",
        "PR30"
    ]
}
},
{
    "index": "patient",
    "doc_type": "Patient",
    "id": "PT08",
    "body": {
        "patient_id": "PT08",
        "personal_data": "Михайлова Елена Сергеевна",
        "voucher": "9876543",
        "date_of_arrival": "2021-02-10",
        "number_of_days": "6",
        "diagnosis": [
            "Почечная недостаточность",
            "Цистит",
            "Острая респираторная вирусная инфекция"
        ],
        "precedures_id": [
            "PR04",
            "PR12",
            "PR17",
            "PR25"
        ]
    }
},
{
    "index": "patient",
    "doc_type": "Patient",
    "id": "PT09",
    "body": {
        "patient_id": "PT09",
        "personal_data": "Смирнов Алексей Владимирович",
        "voucher": "4321567",
        "date_of_arrival": "2018-07-17",
        "number_of_days": "11",
        "diagnosis": [
            "Депрессия",
            "Панические атаки",
            "Психосоматические расстройства"
        ],
        "precedures_id": [
            "PR01",
            "PR03",
            "PR30"
        ]
    }
},
{
    "index": "patient",
    "doc_type": "Patient",
    "id": "PT10",
    "body": {
        "patient_id": "PT10",
        "personal_data": "Иванов Андрей Петрович",
        "voucher": "7654321",
        "date_of_arrival": "2019-04-11",
        "number_of_days": "7",
        "diagnosis": [
            "Бронхит",
            "Астма",
            "Пневмония"
        ],
        "precedures_id": [
            "PR02",
            "PR06",
            "PR15",
            "PR23",
            "PR28"
        ]
    }
},
{

```

```

    "index": "patient",
    "doc_type": "Patient",
    "id": "PT11",
    "body": {
      "patient_id": "PT11",
      "personal_data": "Сергеева Екатерина Дмитриевна",
      "voucher": "8765432",
      "date_of_arrival": "2020-09-30",
      "number_of_days": "9",
      "diagnosis": [
        "Онкологическое заболевание",
        "Лейкемия",
        "Меланома"
      ],
      "precedures_id": [
        "PR01",
        "PR07",
        "PR13",
        "PR22",
        "PR26",
        "PR30"
      ]
    }
  },
  {
    "index": "patient",
    "doc_type": "Patient",
    "id": "PT12",
    "body": {
      "patient_id": "PT12",
      "personal_data": "Петров Сергей Иванович",
      "voucher": "9876543",
      "date_of_arrival": "2021-03-15",
      "number_of_days": "5",
      "diagnosis": [
        "Ишемическая болезнь сердца",
        "Гиперлипидемия",
        "Артериальная гипертензия"
      ],
      "precedures_id": [
        "PR02",
        "PR08",
        "PR14",
        "PR24"
      ]
    }
  },
  {
    "index": "patient",
    "doc_type": "Patient",
    "id": "PT13",
    "body": {
      "patient_id": "PT13",
      "personal_data": "Макарова Ольга Степановна",
      "voucher": "5432109",
      "date_of_arrival": "2018-11-21",
      "number_of_days": "13",
      "diagnosis": [
        "Остеохондроз",
        "Сколиоз",
        "Грыжа позвоночника"
      ],
      "precedures_id": [
        "PR01",
        "PR05",
        "PR11",
        "PR18",
        "PR27",
        "PR29"
      ]
    }
  },
  {
    "index": "patient",
    "doc_type": "Patient",
    "id": "PT14",
    "body": {
      "patient_id": "PT14",
      "personal_data": "Иванова Наталья Сергеевна",
      "voucher": "9870123",

```

```

    "date_of_arrival": "2019-07-17",
    "number_of_days": "8",
    "diagnosis": [
        "Диабет",
        "Ожирение",
        "Гипертиреоз"
    ],
    "precedures_id": [
        "PR02",
        "PR04",
        "PR09",
        "PR25"
    ]
  },
  {
    "index": "patient",
    "doc_type": "Patient",
    "id": "PT15",
    "body": {
      "patient_id": "PT15",
      "personal_data": "Петрова Олеся Михайловна",
      "voucher": "5678901",
      "date_of_arrival": "2020-05-11",
      "number_of_days": "14",
      "diagnosis": [
        "Остеопороз",
        "Ревматоидный артрит",
        "Люпус"
      ],
      "precedures_id": [
        "PR11",
        "PR19",
        "PR27",
        "PR30"
      ]
    }
  },
  {
    "index": "patient",
    "doc_type": "Patient",
    "id": "PT16",
    "body": {
      "patient_id": "PT16",
      "personal_data": "Сидоров Павел Анатольевич",
      "voucher": "2345678",
      "date_of_arrival": "2018-06-28",
      "number_of_days": "6",
      "diagnosis": [
        "Хронический гастрит",
        "Язва желудка",
        "Панкреатит"
      ],
      "precedures_id": [
        "PR02",
        "PR07",
        "PR13",
        "PR20",
        "PR26"
      ]
    }
  },
  {
    "index": "patient",
    "doc_type": "Patient",
    "id": "PT17",
    "body": {
      "patient_id": "PT17",
      "personal_data": "Павлова Юлия Викторовна",
      "voucher": "4567890",
      "date_of_arrival": "2019-10-15",
      "number_of_days": "11",
      "diagnosis": [
        "Депрессия",
        "Психоз",
        "Шизофрения"
      ],
      "precedures_id": [
        "PR03",
        "PR08",

```

```

        "PR30"
    ]
}
},
{
    "index": "patient",
    "doc_type": "Patient",
    "id": "PT18",
    "body": {
        "patient_id": "PT18",
        "personal_data": "Григорьев Владимир Игоревич",
        "voucher": "7654321",
        "date_of_arrival": "2020-02-28",
        "number_of_days": "9",
        "diagnosis": [
            "Простуда",
            "Грипп",
            "ОРВИ"
        ],
        "precedures_id": [
            "PR01",
            "PR06",
            "PR18",
            "PR24",
            "PR30"
        ]
    }
}
},
{
    "index": "patient",
    "doc_type": "Patient",
    "id": "PT19",
    "body": {
        "patient_id": "PT19",
        "personal_data": "Михайлов Алексей Владимирович",
        "voucher": "9876543",
        "date_of_arrival": "2019-03-10",
        "number_of_days": "12",
        "diagnosis": [
            "Острый аппендицит",
            "Гастроэнтерит",
            "Колит"
        ],
        "precedures_id": [
            "PR02",
            "PR05",
            "PR12",
            "PR17",
            "PR22",
            "PR29"
        ]
    }
}
},
{
    "index": "patient",
    "doc_type": "Patient",
    "id": "PT20",
    "body": {
        "patient_id": "PT20",
        "personal_data": "Николаева Елена Васильевна",
        "voucher": "5432109",
        "date_of_arrival": "2018-12-05",
        "number_of_days": "7",
        "diagnosis": [
            "Бронхит",
            "Астма",
            "Пневмония"
        ],
        "precedures_id": [
            "PR03",
            "PR08",
            "PR15",
            "PR20",
            "PR28"
        ]
    }
}
},
{
    "index": "patient",
    "doc_type": "Patient",

```



```

    "id": "PT21",
    "body": {
      "patient_id": "PT21",
      "personal_data": "Смирнов Андрей Андреевич",
      "voucher": "8765432",
      "date_of_arrival": "2020-07-12",
      "number_of_days": "10",
      "diagnosis": [
        "Геморрой",
        "Проктит",
        "Протозоальные инфекции"
      ],
      "precedures_id": [
        "PR01",
        "PR06",
        "PR11",
        "PR17",
        "PR23",
        "PR29"
      ]
    }
  },
  {
    "index": "patient",
    "doc_type": "Patient",
    "id": "PT22",
    "body": {
      "patient_id": "PT22",
      "personal_data": "Поляков Денис Сергеевич",
      "voucher": "2109876",
      "date_of_arrival": "2019-09-20",
      "number_of_days": "15",
      "diagnosis": [
        "Ожирение",
        "Дислипидемия",
        "Метаболический синдром"
      ],
      "precedures id": [
        "PR02",
        "PR07",
        "PR14",
        "PR20",
        "PR26",
        "PR30"
      ]
    }
  },
  {
    "index": "patient",
    "doc_type": "Patient",
    "id": "PT23",
    "body": {
      "patient_id": "PT23",
      "personal_data": "Ковалев Егор Викторович",
      "voucher": "9876543",
      "date_of_arrival": "2018-06-15",
      "number_of_days": "14",
      "diagnosis": [
        "Гепатит",
        "Цирроз печени",
        "Холестистит"
      ],
      "precedures_id": [
        "PR03",
        "PR08",
        "PR12",
        "PR17",
        "PR22",
        "PR28"
      ]
    }
  },
  {
    "index": "patient",
    "doc_type": "Patient",
    "id": "PT24",
    "body": {
      "patient_id": "PT24",
      "personal_data": "Лебедева Ксения Сергеевна",
      "voucher": "5432109",

```

```

      "date_of_arrival": "2019-04-28",
      "number_of_days": "7",
      "diagnosis": [
        "Депрессия",
        "Тревожное расстройство",
        "Психоз"
      ],
      "precedures_id": [
        "PR01",
        "PR07",
        "PR11",
        "PR18",
        "PR24",
        "PR30"
      ]
    },
    {
      "index": "patient",
      "doc_type": "Patient",
      "id": "PT25",
      "body": {
        "patient_id": "PT25",
        "personal_data": "Велякова Ольга Викторовна",
        "voucher": "8765432",
        "date_of_arrival": "2020-02-10",
        "number_of_days": "10",
        "diagnosis": [
          "Ревматоидный артрит",
          "Остеоартрит",
          "Подагра"
        ],
        "precedures_id": [
          "PR02",
          "PR08",
          "PR14",
          "PR19",
          "PR26",
          "PR30"
        ]
      }
    },
    {
      "index": "patient",
      "doc_type": "Patient",
      "id": "PT26",
      "body": {
        "patient_id": "PT26",
        "personal_data": "Горбачев Сергей Михайлович",
        "voucher": "2109876",
        "date_of_arrival": "2019-10-05",
        "number_of_days": "15",
        "diagnosis": [
          "Инфекционный мононуклеоз",
          "Герпетическая инфекция",
          "Вирусный гепатит"
        ],
        "precedures_id": [
          "PR01",
          "PR07",
          "PR13",
          "PR19",
          "PR23",
          "PR29"
        ]
      }
    },
    {
      "index": "patient",
      "doc_type": "Patient",
      "id": "PT27",
      "body": {
        "patient_id": "PT27",
        "personal_data": "Титов Антон Валерьевич",
        "voucher": "9876543",
        "date_of_arrival": "2018-07-20",
        "number_of_days": "14",
        "diagnosis": [
          "Артериальная гипертензия",
          "Ишемическая болезнь сердца",

```

```

        "Острая сердечная недостаточность"
    ],
    "precedures_id": [
        "PR03",
        "PR09",
        "PR12",
        "PR18",
        "PR24",
        "PR28"
    ]
}
},
{
    "index": "patient",
    "doc type": "Patient",
    "id": "PT28",
    "body": {
        "patient_id": "PT28",
        "personal_data": "Иванов Алексей Петрович",
        "voucher": "8765432",
        "date_of_arrival": "2022-01-15",
        "number_of_days": "5",
        "diagnosis": [
            "Острый панкреатит",
            "Желчекаменная болезнь",
            "Гастроэзофагеальная рефлюксная болезнь"
        ],
        "precedures_id": [
            "PR02",
            "PR08",
            "PR14",
            "PR19",
            "PR25",
            "PR30"
        ]
    }
},
{
    "index": "patient",
    "doc_type": "Patient",
    "id": "PT29",
    "body": {
        "patient_id": "PT29",
        "personal_data": "Смирнова Екатерина Дмитриевна",
        "voucher": "5432109",
        "date_of_arrival": "2022-05-28",
        "number_of_days": "10",
        "diagnosis": [
            "Острый бронхит",
            "Астма",
            "Хроническая обструктивная болезнь легких"
        ],
        "precedures_id": [
            "PR01",
            "PR06",
            "PR11",
            "PR18",
            "PR23",
            "PR29"
        ]
    }
},
{
    "index": "patient",
    "doc_type": "Patient",
    "id": "PT30",
    "body": {
        "patient_id": "PT30",
        "personal_data": "Медведев Петр Андреевич",
        "voucher": "9876543",
        "date_of_arrival": "2021-03-10",
        "number_of_days": "7",
        "diagnosis": [
            "Острый гастрит",
            "Дуоденит",
            "Гастроэнтероколит"
        ],
        "precedures_id": [
            "PR03",
            "PR08",

```

```
        "PR12",  
        "PR18",  
        "PR25",  
        "PR30"  
    ]  
}  
]
```

### 3 Результат выполнения первого запроса к кластеру ES

GET patient/\_search

```
{
  "size": 0,
  "aggregations": {
    "year_period": {
      "date_histogram": {
        "field": "date_of_arrival",
        "calendar_interval": "year",
        "format": "yyyy-MM-dd"
      }
    },
    "aggregations": {
      "Procedure": {
        "terms": {
          "field": "precedures_id",
          "order": {
            "_key": "asc"
          }
        }
      },
      "aggregations": {
        "number_of_patients": {
          "value_count": {
            "field": "patient_id"
          }
        }
      }
    }
  }
}
```

---

```
{
  "took" : 115,
  "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" : "2018-01-01",
          "key" : 1514764800000,
          "doc_count" : 8,
          "Procedure" : {
            "doc_count_error_upper_bound" : 0,
            "sum_other_doc_count" : 17,
            "buckets" : [
              {
                "key" : "pr01",
                "doc_count" : 3,

```

```

        "number_of_patients" : {
            "value" : 3
        }
    },
    {
        "key" : "pr02",
        "doc_count" : 1,
        "number_of_patients" : {
            "value" : 1
        }
    },
    {
        "key" : "pr03",
        "doc_count" : 5,
        "number_of_patients" : {
            "value" : 5
        }
    },
    {
        "key" : "pr05",
        "doc_count" : 2,
        "number_of_patients" : {
            "value" : 2
        }
    },
    {
        "key" : "pr07",
        "doc_count" : 2,
        "number_of_patients" : {
            "value" : 2
        }
    },
    {
        "key" : "pr08",
        "doc_count" : 2,
        "number_of_patients" : {
            "value" : 2
        }
    },
    {
        "key" : "pr09",
        "doc_count" : 1,
        "number_of_patients" : {
            "value" : 1
        }
    },
    {
        "key" : "pr11",
        "doc_count" : 1,
        "number_of_patients" : {
            "value" : 1
        }
    },
    {
        "key" : "pr12",
        "doc_count" : 3,
        "number_of_patients" : {
            "value" : 3
        }
    },
    {
        "key" : "pr13",
        "doc_count" : 2,
        "number_of_patients" : {

```

```

        "value" : 2
    }
}
]
}
},
{
    "key_as_string" : "2019-01-01",
    "key" : 1546300800000,
    "doc_count" : 9,
    "Procedure" : {
        "doc_count_error_upper_bound" : 0,
        "sum_other_doc_count" : 25,
        "buckets" : [
            {
                "key" : "pr01",
                "doc_count" : 2,
                "number_of_patients" : {
                    "value" : 2
                }
            },
            {
                "key" : "pr02",
                "doc_count" : 5,
                "number_of_patients" : {
                    "value" : 5
                }
            },
            {
                "key" : "pr03",
                "doc_count" : 1,
                "number_of_patients" : {
                    "value" : 1
                }
            },
            {
                "key" : "pr04",
                "doc_count" : 1,
                "number_of_patients" : {
                    "value" : 1
                }
            },
            {
                "key" : "pr05",
                "doc_count" : 1,
                "number_of_patients" : {
                    "value" : 1
                }
            },
            {
                "key" : "pr06",
                "doc_count" : 1,
                "number_of_patients" : {
                    "value" : 1
                }
            },
            {
                "key" : "pr07",
                "doc_count" : 4,
                "number_of_patients" : {
                    "value" : 4
                }
            }
        ]
    }
}
{

```

```

        "key" : "pr08",
        "doc_count" : 2,
        "number_of_patients" : {
            "value" : 2
        }
    },
    {
        "key" : "pr09",
        "doc_count" : 2,
        "number_of_patients" : {
            "value" : 2
        }
    },
    {
        "key" : "pr11",
        "doc_count" : 1,
        "number_of_patients" : {
            "value" : 1
        }
    }
]
}
},
{
    "key_as_string" : "2020-01-01",
    "key" : 1577836800000,
    "doc_count" : 7,
    "Procedure" : {
        "doc_count_error_upper_bound" : 0,
        "sum_other_doc_count" : 20,
        "buckets" : [
            {
                "key" : "pr01",
                "doc_count" : 3,
                "number_of_patients" : {
                    "value" : 3
                }
            },
            {
                "key" : "pr02",
                "doc_count" : 2,
                "number_of_patients" : {
                    "value" : 2
                }
            },
            {
                "key" : "pr04",
                "doc_count" : 1,
                "number_of_patients" : {
                    "value" : 1
                }
            },
            {
                "key" : "pr06",
                "doc_count" : 2,
                "number_of_patients" : {
                    "value" : 2
                }
            },
            {
                "key" : "pr07",
                "doc_count" : 1,
                "number_of_patients" : {
                    "value" : 1
                }
            }
        ]
    }
}

```



```

    }
  },
  {
    "key" : "pr08",
    "doc_count" : 2,
    "number_of_patients" : {
      "value" : 2
    }
  },
  {
    "key" : "pr10",
    "doc_count" : 1,
    "number_of_patients" : {
      "value" : 1
    }
  },
  {
    "key" : "pr11",
    "doc_count" : 2,
    "number_of_patients" : {
      "value" : 2
    }
  },
  {
    "key" : "pr13",
    "doc_count" : 1,
    "number_of_patients" : {
      "value" : 1
    }
  },
  {
    "key" : "pr14",
    "doc_count" : 1,
    "number_of_patients" : {
      "value" : 1
    }
  }
]
}
},
{
  "key_as_string" : "2021-01-01",
  "key" : 1609459200000,
  "doc_count" : 4,
  "Procedure" : {
    "doc_count_error_upper_bound" : 0,
    "sum_other_doc_count" : 7,
    "buckets" : [
      {
        "key" : "pr02",
        "doc_count" : 1,
        "number_of_patients" : {
          "value" : 1
        }
      },
      {
        "key" : "pr03",
        "doc_count" : 1,
        "number_of_patients" : {
          "value" : 1
        }
      },
      {
        "key" : "pr04",

```

```

        "doc_count" : 1,
        "number_of_patients" : {
            "value" : 1
        }
    },
    {
        "key" : "pr06",
        "doc_count" : 1,
        "number_of_patients" : {
            "value" : 1
        }
    },
    {
        "key" : "pr08",
        "doc_count" : 2,
        "number_of_patients" : {
            "value" : 2
        }
    },
    {
        "key" : "pr09",
        "doc_count" : 1,
        "number_of_patients" : {
            "value" : 1
        }
    },
    {
        "key" : "pr11",
        "doc_count" : 1,
        "number_of_patients" : {
            "value" : 1
        }
    },
    {
        "key" : "pr12",
        "doc_count" : 2,
        "number_of_patients" : {
            "value" : 2
        }
    },
    {
        "key" : "pr14",
        "doc_count" : 1,
        "number_of_patients" : {
            "value" : 1
        }
    },
    {
        "key" : "pr17",
        "doc_count" : 1,
        "number_of_patients" : {
            "value" : 1
        }
    }
]
},
{
    "key_as_string" : "2022-01-01",
    "key" : 1640995200000,
    "doc_count" : 2,
    "Procedure" : {
        "doc_count_error_upper_bound" : 0,
        "sum_other_doc_count" : 2,

```

```

"buckets" : [
  {
    "key" : "pr01",
    "doc_count" : 1,
    "number_of_patients" : {
      "value" : 1
    }
  },
  {
    "key" : "pr02",
    "doc_count" : 1,
    "number_of_patients" : {
      "value" : 1
    }
  },
  {
    "key" : "pr06",
    "doc_count" : 1,
    "number_of_patients" : {
      "value" : 1
    }
  },
  {
    "key" : "pr08",
    "doc_count" : 1,
    "number_of_patients" : {
      "value" : 1
    }
  },
  {
    "key" : "pr11",
    "doc_count" : 1,
    "number_of_patients" : {
      "value" : 1
    }
  },
  {
    "key" : "pr14",
    "doc_count" : 1,
    "number_of_patients" : {
      "value" : 1
    }
  },
  {
    "key" : "pr18",
    "doc_count" : 1,
    "number_of_patients" : {
      "value" : 1
    }
  },
  {
    "key" : "pr19",
    "doc_count" : 1,
    "number_of_patients" : {
      "value" : 1
    }
  },
  {
    "key" : "pr23",
    "doc_count" : 1,
    "number_of_patients" : {
      "value" : 1
    }
  },
],

```



```

        "Процедура, основанная на использовании лечебных <em>трав</em> и
растений для оздоровления организма, укрепления"
    ]
}
},
{
    "_index" : "procedure",
    "_type" : "Procedure",
    "_id" : "PR25",
    "_score" : 2.183346,
    "_source" : {
        "price" : "6500",
        "name" : "Фитотерапия"
    },
    "highlight" : {
        "description" : [
            "Процедура, основанная на использовании лечебных растений и
<em>трав</em> для оздоровления организма, укрепления"
        ]
    }
},
{
    "_index" : "procedure",
    "_type" : "Procedure",
    "_id" : "PR14",
    "_score" : 2.062953,
    "_source" : {
        "price" : "8500",
        "name" : "Фитобочка"
    },
    "highlight" : {
        "description" : [
            "Процедура, основанная на воздействии на организм паром с
добавлением лечебных <em>трав</em>, что способствует"
        ]
    }
}
]
}
}

```

## **ПРИЛОЖЕНИЕ Б**

### Исходные коды программ

# 1 Программа маппинга и индексации документов типа «Процедура»

## и «Пациент»

```
import json
from elasticsearch import Elasticsearch

client = Elasticsearch([{"host": "127.0.0.1", "port": 9200}])
index_1 = "procedure"
index_2 = "patient"

client.indices.delete(index=index_1)
client.indices.delete(index=index_2)
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": {
            "analytic_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)
```

```

ProcedureMapping = {
    "properties":{
        "name": {
            "type": "text",
            "fielddata": True,
            "analyzer":"analitic_for_ru",
            "search_analyzer":"analitic_for_ru"
        },
        "description": {
            "type": "text",
            "fielddata": True,
            "analyzer":"analitic_for_ru",
            "search_analyzer":"analitic_for_ru"
        },
        "price": {
            "type": "integer"
        }
    }
}

client.indices.put_mapping(index=index_1,
                           doc_type="Procedure",
                           include_type_name="true",
                           body=ProcedureMapping)

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

with open("Procedure.json", 'r') as file_Procedure:
    Procedure_data = json.load(file_Procedure)

for data in Procedure_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("Procedure_indexed")

client.indices.open(index=index_2)

PatientMapping = {
    "properties":{
        "patient_id": {
            "type": "text",
            "fielddata": True
        },
        "personal_data": {
            "type": "text",
            "analyzer":"analitic_for_ru",
            "search_analyzer":"analitic_for_ru",
            "fielddata": True
        },
        "voucher": {
            "type": "text",
            "fielddata": True
        },
        "date_of_arrival": {

```



```

        "type": "date",
        "format": "yyyy-MM-dd"
    },
    "number_of_days": {
        "type": "integer"
    },
    "diagnosis":{
        "type": "text",
        "analyzer":"analitic_for_ru",
        "search_analyzer":"analitic_for_ru",
        "fielddata": True
    },
    "precedures_id": {
        "type": "text",
        "fielddata": True
    }
}

}

client.indices.put_mapping(index=index_2,
                           doc_type="Patient",
                           include_type_name="true",
                           body=PatientMapping)

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

with open("Patient.json", 'r') as file_Client:
    data_Client = json.load(file_Client)

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

```

## 2 Тело первого запроса к кластеру ES

```
GET patient/_search
{
  "size": 0,
  "aggregations": {
    "year_period": {
      "date_histogram": {
        "field": "date_of_arrival",
        "calendar_interval": "year",
        "format": "yyyy-MM-dd"
      },
    },
    "aggregations": {
      "Procedure": {
        "terms": {
          "field": "precedures_id",
          "order": {
            "_key": "asc"
          }
        },
      },
      "aggregations": {
        "number_of_patients": {
          "value_count": {
            "field": "patient_id"
          }
        }
      }
    }
  }
}
```

---

## 3 Тело второго запроса к кластеру ES

```
GET procedure/_search
{
  "query": {
    "simple_query_string": {
      "query": "Травы"
    }
  },
  "_source": [
    "name",
    "price"
  ],
  "highlight": {
    "fields": {
      "description": {}
    }
  }
}
```

## 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 = "procedure"
indexName2 = "patient"

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

graph_db.delete_all()

#array of procedure
procedure_array = {
    "size": 1000,
    "query": {
        "match_all": {}
    }
}

#array of clients
patient_array = {
    "size": 1000,
    "query": {
        "match_all": {}
    }
}

result_procedure = client.search(index = indexName1, body = procedure_array)
result_patient = client.search(index = indexName2, body = patient_array)

# cycle for my_node
for my_node in result_patient ['hits']['hits']:
    Patient_Node = Node("patient",
        Patient_id=my_node['_source']['patient_id'],
        Patient_personal_data=my_node['_source']['personal_data'],
        Voucher=my_node['_source']['voucher'],
        Arrival_date=my_node['_source']['date_of_arrival'],
        Numb_of_days=my_node['_source']['number_of_days'],
        Diagnosis=my_node['_source']['diagnosis'],
        Procedures_id=my_node['_source']['precedures_id'])
    graph_db.create(Patient_Node)
    try:
        for procedure in result_procedure ['hits']['hits']:
            Procedure_Node = graph_db.nodes.match("Procedure",
                Procedure_id=procedure['_id']).first()
            if Procedure_Node == None:
                Procedure_Node = Node("Procedure",
                    Procedure_id=procedure['_id'],
                    Name = procedure['_source']['name'],
                    Description=procedure['_source']['description'],
                    Price = procedure['_source']['price'])
                graph_db.create(Procedure_Node)
            # Link
            if procedure['_id'] in my_node['_source']['precedures_id']:
                Got_procedure = Relationship( Patient_Node,
                    "Got_procedure", Procedure_Node,
                    Procedure_Price=procedure['_source']['price'])
                graph_db.create(Got_procedure)

    except Exception as e:
        print(e)
```

## 5 Тело запроса запроса к базе данных Neo4j

```
MATCH p=(pct:patient)-[r:Got_procedure]->(proc:Procedure)
WITH proc, sum(toInteger(r.Procedure_Price)) as procedure_sum, count(r) as
number_of_client
ORDER BY procedure_sum desc
RETURN proc.Name, procedure_sum, proc.Price, number_of_client, proc.Description
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
#import findspark
#findspark.init()

client = Elasticsearch([{"host": "localhost", "port": 9200}])
indexName1 = "procedure"
indexName2 = "patient"

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

result_procedure = client.search(index=indexName1,
body=searchBody)['hits']['hits']
result_patient = client.search(index=indexName2, body=searchBody)['hits']['hits']
result_naznachenie = client.search(index=indexName2,
body=searchBody)['hits']['hits']

# schema for procedure, patient and naznachenie
Schema_for_procedure = StructType([
    StructField("procedure_id", StringType(), True),
    StructField("name", StringType(), True),
    StructField("description", StringType(), True),
    StructField("price", IntegerType(), True),
])

Schema_for_patient = StructType([
    StructField("patient_id", StringType(), True),
    StructField("personal_data", StringType(), True),
    StructField("voucher", StringType(), True),
    StructField("diagnosis", StringType(), True)
])

Schema_for_naznachenie= StructType([
    StructField("voucher", StringType(), True),
    StructField("procedures_id", StringType(), True),
])

# procedure table
Table_for_procedure_data = []
for procedure in result_procedure:
    Table_for_procedure_data.append((
        procedure['_id'],
        procedure['_source']['name'],
        procedure['_source']['description'],
        int(procedure['_source']['price'])
    ))

# patient table
Table_for_patient_data = []
for patient in result_patient:
    Table_for_patient_data.append((
        patient['_source']['patient_id'],
        patient['_source']['personal_data'],
        patient['_source']['voucher'],
        patient['_source']['diagnosis'],
```

```

    ))

# naznachenie table
Table_for_naznachenie_data = []
for naznachenie in result_naznachenie:
    Table_for_naznachenie_data.append((
        naznachenie['_source']['voucher'],
        naznachenie['_source']['precedures_id']
    ))

#Creating data frame
Table_for_procedure = sparkSession.createDataFrame(Table_for_procedure_data,
Schema_for_procedure)
Table_for_patient = sparkSession.createDataFrame(Table_for_patient_data,
Schema_for_patient)
Table_for_naznachenie = sparkSession.createDataFrame(Table_for_naznachenie_data,
Schema_for_naznachenie)

# making csv
Table_for_procedure.write.csv(path='hdfs://localhost:9000/procedure.csv',
mode='overwrite', header=True)
Table_for_patient.write.csv(path='hdfs://localhost:9000/patient.csv',
mode='overwrite', header=True)
Table_for_naznachenie.write.csv(path='hdfs://localhost:9000/naznachenie.csv',
mode='overwrite', header=True)

df_load = sparkSession.read.load(path='hdfs://localhost:9000/procedure.csv',
format='csv', sep=',', inferSchema="true", header="true")
df_load.show()
df_load = sparkSession.read.load(path='hdfs://localhost:9000/patient.csv',
format='csv', sep=',', inferSchema="true", header="true")
df_load.show()
df_load = sparkSession.read.load(path='hdfs://localhost:9000/naznachenie.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
from pyspark.sql.functions import array_contains
from pyspark.sql.types import StringType
from pyspark.sql.functions import udf, expr, concat, col

sparkSession=SparkSession.builder.appName("spark-
csv").config("spark.some.config.option", "5").getOrCreate()

Procedure_Table =
sparkSession.read.load(path='hdfs://localhost:9000/procedure.csv', format='csv',
sep=',', inferSchema="true", header="true")
Naznachenie_Table =
sparkSession.read.load(path='hdfs://localhost:9000/naznachenie.csv',
format='csv', sep=',', inferSchema="true", header="true")

Procedure_Table.registerTempTable("procedure")
Naznachenie_Table.registerTempTable("naznachenie")

df = sparkSession.sql("SELECT ID_OF_PROCEDURE, COUNT(ID_OF_PROCEDURE) as
NUMBER_OF_NAZNACH FROM (SELECT procedure.procedure_id as ID_OF_PROCEDURE,
naznachenie.procedures_id LIKE concat(concat('%', procedure.procedure_id),'%')
as CHECK FROM procedure, naznachenie) WHERE check=true GROUP BY ID_OF_PROCEDURE
ORDER BY ID_OF_PROCEDURE").show(40)

input('Ctrl C')
```

## 8 Физический план в мониторе Spark

```
== Parsed Logical Plan ==
GlobalLimit 21
+- LocalLimit 21
  +- Project [cast(ID_OF_PROCEDURE#44 as string) AS ID_OF_PROCEDURE#53,
cast(NUMBER_OF_NAZNACH#46L as string) AS NUMBER_OF_NAZNACH#54]
    +- Project [ID_OF_PROCEDURE#44, NUMBER_OF_NAZNACH#46L]
      +- Sort [ID_OF_PROCEDURE#44 ASC NULLS FIRST], true
      +- Aggregate [ID_OF_PROCEDURE#44], [ID_OF_PROCEDURE#44,
count(ID_OF_PROCEDURE#44) AS NUMBER_OF_NAZNACH#46L]
        +- Filter (check#45 = true)
          +- SubqueryAlias __auto_generated_subquery_name
            +- Project [procedure_id#16 AS ID_OF_PROCEDURE#44,
procedures_id#41 LIKE concat(concat(%, procedure_id#16), %) AS CHECK#45]
              +- Join Inner
                :- SubqueryAlias procedure
                : +-
Relation[procedure_id#16,name#17,description#18,price#19] csv
              +- SubqueryAlias naznachenie
                +- Relation[voucher#40,procedures_id#41] csv

== Analyzed Logical Plan ==
ID_OF_PROCEDURE: string, NUMBER_OF_NAZNACH: string
GlobalLimit 21
+- LocalLimit 21
  +- Project [cast(ID_OF_PROCEDURE#44 as string) AS ID_OF_PROCEDURE#53,
cast(NUMBER_OF_NAZNACH#46L as string) AS NUMBER_OF_NAZNACH#54]
    +- Project [ID_OF_PROCEDURE#44, NUMBER_OF_NAZNACH#46L]
      +- Sort [ID_OF_PROCEDURE#44 ASC NULLS FIRST], true
      +- Aggregate [ID_OF_PROCEDURE#44], [ID_OF_PROCEDURE#44,
count(ID_OF_PROCEDURE#44) AS NUMBER_OF_NAZNACH#46L]
        +- Filter (check#45 = true)
          +- SubqueryAlias __auto_generated_subquery_name
            +- Project [procedure_id#16 AS ID_OF_PROCEDURE#44,
procedures_id#41 LIKE concat(concat(%, procedure_id#16), %) AS CHECK#45]
              +- Join Inner
                :- SubqueryAlias procedure
                : +-
Relation[procedure_id#16,name#17,description#18,price#19] csv
              +- SubqueryAlias naznachenie
                +- Relation[voucher#40,procedures_id#41] csv

== Optimized Logical Plan ==
GlobalLimit 21
+- LocalLimit 21
  +- Project [ID_OF_PROCEDURE#44, cast(NUMBER_OF_NAZNACH#46L as string) AS
NUMBER_OF_NAZNACH#54]
    +- Sort [ID_OF_PROCEDURE#44 ASC NULLS FIRST], true
    +- Aggregate [ID_OF_PROCEDURE#44], [ID_OF_PROCEDURE#44,
count(ID_OF_PROCEDURE#44) AS NUMBER_OF_NAZNACH#46L]
      +- Project [procedure_id#16 AS ID_OF_PROCEDURE#44]
      +- Join Inner, (procedures_id#41 LIKE concat(concat(%,
procedure_id#16), %) = true)
        :- Project [procedure_id#16]
        : +- Relation[procedure_id#16,name#17,description#18,price#19]
csv
      +- Project [procedures_id#41]
      +- Filter isnotnull(procedures_id#41)
      +- Relation[voucher#40,procedures_id#41] csv

== Physical Plan ==
TakeOrderedAndProject(limit=21, orderBy=[ID_OF_PROCEDURE#44 ASC NULLS FIRST],
output=[ID_OF_PROCEDURE#44,NUMBER_OF_NAZNACH#54])
```



```

+- *(3) HashAggregate(keys=[ID_OF_PROCEDURE#44],
functions=[count(ID_OF_PROCEDURE#44)], output=[ID_OF_PROCEDURE#44,
NUMBER_OF_NAZNACH#46L])
  +- Exchange hashpartitioning(ID_OF_PROCEDURE#44, 200), true, [id=#91]
    +- *(2) HashAggregate(keys=[ID_OF_PROCEDURE#44],
functions=[partial_count(ID_OF_PROCEDURE#44)], output=[ID_OF_PROCEDURE#44,
count#58L])
      +- *(2) Project [procedure_id#16 AS ID_OF_PROCEDURE#44]
        +- BroadcastNestedLoopJoin BuildRight, Inner, (procedures_id#41 LIKE
concat(concat(%, procedure_id#16), %) = true)
          :- FileScan csv [procedure_id#16] Batched: false, DataFilters: [],
Format: CSV, Location: InMemoryFileIndex[hdfs://localhost:9000/procedure.csv],
PartitionFilters: [], PushedFilters: [], ReadSchema: struct<procedure_id:string>
            +- BroadcastExchange IdentityBroadcastMode, [id=#85]
              +- *(1) Project [procedures_id#41]
                +- *(1) Filter isnotnull(procedures_id#41)
                  +- FileScan csv [procedures_id#41] Batched: false,
DataFilters: [isnotnull(procedures_id#41)], Format: CSV, Location:
InMemoryFileIndex[hdfs://localhost:9000/naznachenie.csv], PartitionFilters: [],
PushedFilters: [IsNotNull(procedures_id)], ReadSchema:
struct<procedures_id: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.