**ТИТУЛ**

**ЗАДАНИЕ**

**РЕФЕРАТ**

**СОДЕРЖАНИЕ**

[Определения, обозначения и сокращения 5](#_Toc1)

[Введение 6](#_Toc2)

[1 Задание и описание варианта КП 7](#_Toc3)

[2 Установка виртуальной машины, установка пакетов Elasticsearch, Neo4j, Hadoop+Spark 8](#_Toc4)

[2.1 Развертывание виртуальной машины 9](#_Toc5)

[2.2 Установка ElasticSearch 10](#_Toc6)

[2.3 Установка Neo4j 11](#_Toc7)

[2.4 Установка Hadoop+Spark 12](#_Toc8)

[3 Elasticsearch 13](#_Toc9)

[3.1 Индексация документов 14](#_Toc10)

[3.2 Запросы 16](#_Toc11)

[4 Neo4j 18](#_Toc12)

[4.1 Создание и заполнение графовой БД 18](#_Toc13)

[4.2 Запрос 20](#_Toc14)

[5 Spark 21](#_Toc15)

[5.1 Создание и заполнение таблиц 22](#_Toc16)

[5.2 Запрос 23](#_Toc17)

[5.3 Монитор 24](#_Toc18)

[Заключение 25](#_Toc19)

[Список использованных источников 26](#_Toc20)

[ПРИЛОЖЕНИЕ А 27](#_Toc21)

[ПРИЛОЖЕНИЕ Б 37](#_Toc22)

[ПРИЛОЖЕНИЕ В 47](#_Toc23)

# **Определения, обозначения и сокращения**

# **Введение**

# **1 Задание и описание варианта КП**

# **2 Установка виртуальной машины, установка пакетов Elasticsearch, Neo4j, Hadoop+Spark**

## **2.1 Развертывание виртуальной машины**

## **2.2 Установка ElasticSearch**

## **2.3 Установка Neo4j**

## **2.4 Установка Hadoop+Spark**

# **3 Elasticsearch**

## **3.1 Индексация документов**

Прежде чем выполнить индексацию документов было необходимо сгенерировать исходные данные. Для этого была использована python-библиотека Faker, позволяющая генерировать ФИО, тексты, даты и другие данные с заданными параметрами [ссылка]**.** В результате были сгенерированы данные со структурами, приведенными ниже**.**

Структура для записи о мастере:

* master\_id — уникальный номер мастера (пример: 163543);
* master\_desc — описание мастера — ФИО и данные об образовании/опыте (Пример: “Максимильян Виленович Гурьев, Стаж Работы: 11 Л./Г..”);
* master\_feedbacks — массив отзывов о мастере (пример: [“медлительный, аккуратный.”, “медлительный, ворчливый.”]).

Структура записи о заказе:

* order\_id — уникальный номер заказа (пример: 405064);
* order\_date — дата получения заказа (пример: 2024-03-15);
* order\_customer\_id — уникальный номер заказчика (пример: 257458);
* order\_customer\_desc — описание заказчика — ФИО и доп. информация (пример: “Имя: Комарова Дарья Борисовна; постоянный, премиальный.”);
* order\_details\_desc — свойства заказа (пример: “хрупкий, тяжелый.”);
* order\_due\_date — срок выполнения заказа (пример: 2024-04-20);
* order\_fact\_completion\_date — фактическая дата исполнения заказа (пример: 2024-04-25);
* order\_parts — массив со списком необходимых запчастей (пример: [“дисплей”, “usb-разъем”]);
* repair\_types — тип работ (пример: “мелкие детали”, “пайка”);
* order\_price — стоимость заказа (пример: 3866.44);
* order\_master\_id — уникальный номер мастера-исполнителя (пример: 163543)**.**

**ПРОДОЛЖЕНИЕ СЛЕДУЕТ**

## **3.2 Запросы**

Первый запрос должен разбить заказы по дате заказа с периодом 1 месяц, для каждой «корзины» определить суммарное число заказов по каждой запчасти.

Итоговый запрос приведен в листинге Х.

Листинг Х — Запрос №1

|  |
| --- |
| GET order/\_search **# поиск по заказам**  {  **# внешняя агрегация**  "aggs": {  **# название внешней агрегации**  "over\_months": {  **# тип агрегации date histogram**  "date\_histogram": {  "field": "order\_date", **# по полю order date**  "calendar\_interval": "month", **# по месяцам**  "format": "yyyy-MM-dd" **# формат дат**  },  **# внутренняя агрегация**  "aggs":{  **# название внутренней агрегации**  "over\_parts": {  "terms": {  **# по полю order parts**  "field": "order\_parts.keyword"  }  }  }  }  }  } |

Таким образом, внутренняя агрегация выполняет подсчет повторений для каждого типа запчастей, внешняя — организует внутреннюю агрегацию по датам заказа (с округлением до месяцев). Было использовано поле keyword, т.к. без него анализатор разобьет названия некоторых типов запчастей (например, “usb-разъем”)на несколько частей.

Второй запрос должен вывести информацию о мастерах, в сведениях о которых указан стаж работы. Данный запрос приведен в листинге Х.

Листинг Х — запрос №2

|  |
| --- |
| GET master/\_search **# поиск по мастерам**  {  "query": {  "match": { **# поиск записей с вхождением подстроки**  "master\_desc": "стаж" **# "стаж" в поле master\_desc**  }  }  } |

Данный запрос позволяет выполнить поиск записей с вхождением заданной подстроки. Исходный текст документа проходит через анализатор перед выполнением сравнения.

Результаты выполнения запросов приведены в приложении А.

# **4 Neo4j**

## **4.1 Создание и заполнение графовой БД**

Согласно заданию необходимо по данным из Elasticsearch заполнить графовую базу данных узлами вида Заказ(id\_заказа, дата\_заказа, сведения\_о\_заказчике, стоимость\_заказа), Мастер(id\_мастера, сведения\_о\_мастере). А также, внести в Neo4j отношения вида Выполнил(срок\_выполнения\_заказа, фактическая\_дата\_выполнения).



Рисунок Х — схема алгоритма заполнения графовой базы данных

Для заполнения графовой базы данных был разработан алгоритм, схема которого приведена на рисунке Х.

Согласно данному алгоритму сначала выполняется подключение к ES, Neo4j и очистка Neo4j. Затем выполняется запрос, получающий все записи из индекса мастеров.

Для каждого мастера создается узел, затем во вложенном цикле производится поиск соответствующих мастеру заказов и создание узлов заказов и отношений между мастером и его заказами.

Перейдя в веб-интерфейс Neo4j можно увидеть визуализацию созданного графа (рисунок Х).

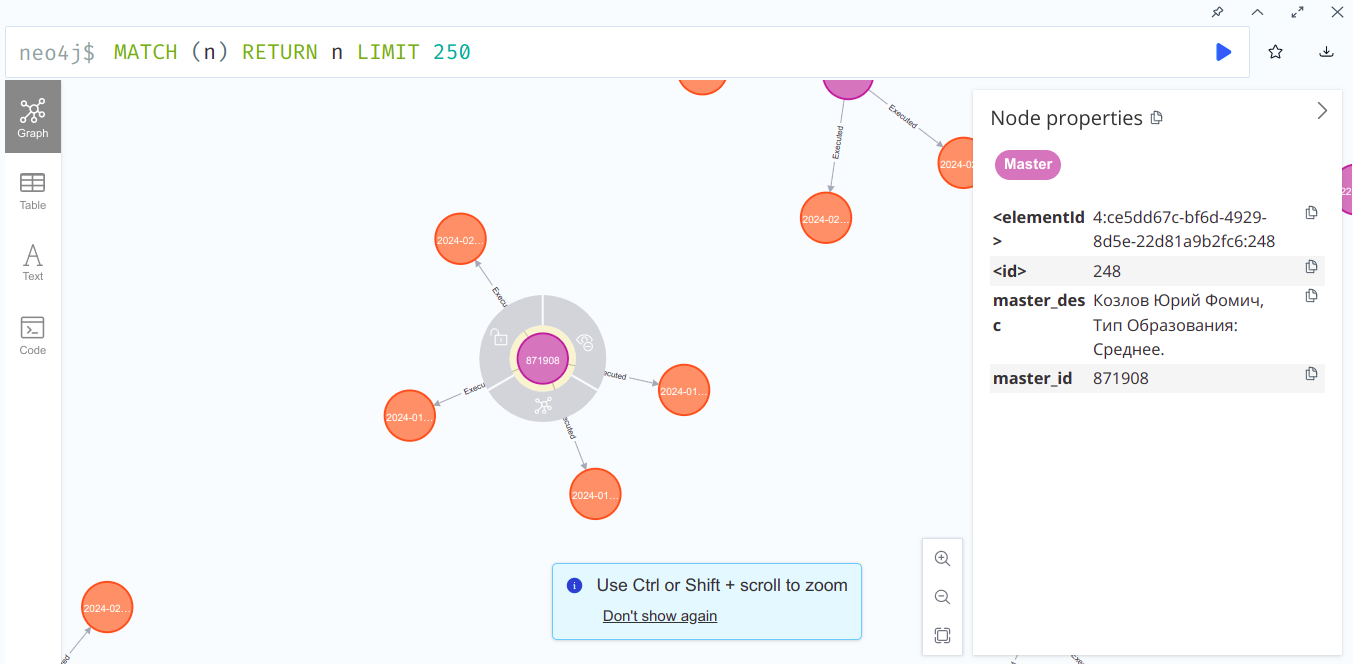


Рисунок Х — визуализация созданного графа

Исходный код программы заполнения графовой базы данных приведен в приложении Б.

## **4.2 Запрос**

В ходе выполнения курсовой работы для Neo4j было необходимо разработать и реализовать запрос: найти мастера, который выполнил максимальное количество заказов.

Для решения данной задачи был написан запрос на языке Cypher. Cypher является декларативным графовым языком запросов, который позволяет писать выразительные и эффективные запросы на получение данных из хранилища графов и их изменение. Cypher является относительно простым, но весьма мощным языком.

Исходный код данного запроса приведен в листинге Х.

Листинг Х — исходный код запроса

|  |
| --- |
| **#соединить узлы мастеров и заказов через отношения “выполнил”**  MATCH (MAS:Master)-[r:Executed]->(ORD:Order)  **# посчитать количество заказов на мастера**  WITH MAS, count(r) AS num  **# вернуть описания мастеров и кол-во заказов**  RETURN MAS.master\_desc as master\_desc, num  **# по убыванию числа заказов**  ORDER BY num DESC  **# лимит 1**  LIMIT 1;  **# результат:**  **# 'Воробьев Ювеналий Измаилович, Тип Образования: Среднее.' 10** |

Данный запрос аналогичен join мастеров и заказов, с последующей группировкой по мастерам и сортировкой в обычном SQL.

# **5 Spark**

## **5.1 Создание и заполнение таблиц**

## **5.2 Запрос**

## **5.3 Монитор**

# **Заключение**

# **Список использованных источников**

# **ПРИЛОЖЕНИЕ А**

Фрагменты JSON и результаты выполнения запросов

Листов 9

**1. Фрагменты JSON с сущностями “Мастер” (первые 10 записей)**

|  |
| --- |
| [  {  "index": "master",  "id": 563268,  "body": {  "master\_id": 563268,  "master\_desc": "Милица Руслановна Фомина, Тип Образования: Среднее.",  "master\_feedbacks": [  "ворчливый, медлительный.",  "ворчливый, медлительный.",  "медлительный, ворчливый."  ]  }  },  {  "index": "master",  "id": 871908,  "body": {  "master\_id": 871908,  "master\_desc": "Козлов Юрий Фомич, Тип Образования: Среднее.",  "master\_feedbacks": [  "ворчливый, аккуратный.",  "медлительный, аккуратный.",  "аккуратный, ворчливый."  ]  }  },  {  "index": "master",  "id": 164475,  "body": {  "master\_id": 164475,  "master\_desc": "Юлий Дмитриевич Константинов, Тип Образования: Среднее.",  "master\_feedbacks": [  "аккуратный, ворчливый."  ]  }  },  {  "index": "master",  "id": 464209,  "body": {  "master\_id": 464209,  "master\_desc": "Ия Тарасовна Степанова, Тип Образования: Высшее.",  "master\_feedbacks": [  "аккуратный, медлительный."  ]  }  },  {  "index": "master",  "id": 22653,  "body": {  "master\_id": 22653,  "master\_desc": "Акулина Рудольфовна Никитина, Стаж Работы: 15 Л./Г..",  "master\_feedbacks": [  "ворчливый, аккуратный.",  "ворчливый, медлительный."  ]  }  },  {  "index": "master",  "id": 282806,  "body": {  "master\_id": 282806,  "master\_desc": "Маслов Емельян Дмитриевич, Тип Образования: Высшее.",  "master\_feedbacks": [  "ворчливый, медлительный."  ]  }  },  {  "index": "master",  "id": 736774,  "body": {  "master\_id": 736774,  "master\_desc": "Владимирова Анжелика Эдуардовна, Тип Образования: Среднее.",  "master\_feedbacks": [  "аккуратный, ворчливый."  ]  }  },  {  "index": "master",  "id": 300714,  "body": {  "master\_id": 300714,  "master\_desc": "Вероника Петровна Силина, Стаж Работы: 12 Л./Г..",  "master\_feedbacks": [  "ворчливый, медлительный.",  "ворчливый, аккуратный."  ]  }  },  {  "index": "master",  "id": 405064,  "body": {  "master\_id": 405064,  "master\_desc": "Хохлов Олег Харлампьевич, Стаж Работы: 3 Л./Г..",  "master\_feedbacks": [  "ворчливый, медлительный.",  "ворчливый, аккуратный.",  "ворчливый, аккуратный."  ]  }  },  {  "index": "master",  "id": 257458,  "body": {  "master\_id": 257458,  "master\_desc": "Шестакова Наина Владиславовна, Стаж Работы: 9 Л./Г..",  "master\_feedbacks": [  "аккуратный, ворчливый.",  "ворчливый, аккуратный."  ]  }  },  ...  ] |

**2. Фрагменты JSON с сущностями “Заказ” (первые 10 записей)**

|  |
| --- |
| [  {  "index": "order",  "id": 131919,  "body": {  "order\_id": 131919,  "order\_date": "2024-01-06",  "order\_customer\_id": 227431,  "order\_customer\_desc": "Имя: Маргарита Валентиновна Чернова; премиальный, постоянный.",  "order\_details\_desc": "хрупкий, ценный.",  "order\_due\_date": "2024-04-10",  "order\_fact\_completion\_date": "2024-04-27",  "order\_parts": [  "аккумулятор"  ],  "repair\_types": [  "замена компонентов"  ],  "order\_price": 2296.89215138,  "order\_master\_id": 257458  }  },  {  "index": "order",  "id": 664730,  "body": {  "order\_id": 664730,  "order\_date": "2024-02-11",  "order\_customer\_id": 656432,  "order\_customer\_desc": "Имя: Комарова Дарья Борисовна; постоянный, премиальный.",  "order\_details\_desc": "хрупкий, тяжелый.",  "order\_due\_date": "2024-04-30",  "order\_fact\_completion\_date": "2024-04-20",  "order\_parts": [  "дисплей",  "дисплей",  "usb-разъем"  ],  "repair\_types": [  "мелкие детали",  "пайка"  ],  "order\_price": 3866.441,  "order\_master\_id": 785701  }  },  {  "index": "order",  "id": 141550,  "body": {  "order\_id": 141550,  "order\_date": "2024-01-10",  "order\_customer\_id": 683623,  "order\_customer\_desc": "Имя: Лора Борисовна Жукова; должник, постоянный.",  "order\_details\_desc": "хрупкий, тяжелый.",  "order\_due\_date": "2024-04-03",  "order\_fact\_completion\_date": "2024-04-27",  "order\_parts": [  "дисплей"  ],  "repair\_types": [  "анализ неисправности",  "мелкие детали"  ],  "order\_price": 4879.1271000384,  "order\_master\_id": 814777  }  },  {  "index": "order",  "id": 99676,  "body": {  "order\_id": 99676,  "order\_date": "2023-12-25",  "order\_customer\_id": 444528,  "order\_customer\_desc": "Имя: Василий Эдуардович Жданов; должник, премиальный.",  "order\_details\_desc": "тяжелый, ценный.",  "order\_due\_date": "2024-04-20",  "order\_fact\_completion\_date": "2024-04-04",  "order\_parts": [  "usb-разъем"  ],  "repair\_types": [  "замена компонентов",  "замена компонентов"  ],  "order\_price": 2103.6634502,  "order\_master\_id": 397809  }  },  {  "index": "order",  "id": 277607,  "body": {  "order\_id": 277607,  "order\_date": "2024-02-22",  "order\_customer\_id": 972535,  "order\_customer\_desc": "Имя: Гурий Анисимович Евдокимов; премиальный, постоянный.",  "order\_details\_desc": "хрупкий, ценный.",  "order\_due\_date": "2024-04-22",  "order\_fact\_completion\_date": "2024-04-10",  "order\_parts": [  "дисплей",  "корпус",  "корпус"  ],  "repair\_types": [  "замена компонентов",  "пайка"  ],  "order\_price": 5652.6299329,  "order\_master\_id": 163543  }  },  {  "index": "order",  "id": 832420,  "body": {  "order\_id": 832420,  "order\_date": "2024-03-01",  "order\_customer\_id": 872571,  "order\_customer\_desc": "Имя: Тимофеев Савва Григорьевич; постоянный, премиальный.",  "order\_details\_desc": "тяжелый, ценный.",  "order\_due\_date": "2024-04-14",  "order\_fact\_completion\_date": "2024-04-11",  "order\_parts": [  "usb-разъем",  "аккумулятор"  ],  "repair\_types": [  "мелкие детали"  ],  "order\_price": 5427.53678234973,  "order\_master\_id": 405064  }  },  {  "index": "order",  "id": 880172,  "body": {  "order\_id": 880172,  "order\_date": "2024-02-08",  "order\_customer\_id": 460465,  "order\_customer\_desc": "Имя: Лыткин Игнатий Валерьянович; должник, премиальный.",  "order\_details\_desc": "тяжелый, объемный.",  "order\_due\_date": "2024-04-23",  "order\_fact\_completion\_date": "2024-04-20",  "order\_parts": [  "usb-разъем"  ],  "repair\_types": [  "пайка",  "замена компонентов"  ],  "order\_price": 7003.855612,  "order\_master\_id": 871908  }  },  {  "index": "order",  "id": 993858,  "body": {  "order\_id": 993858,  "order\_date": "2024-01-16",  "order\_customer\_id": 433504,  "order\_customer\_desc": "Имя: Лукьян Венедиктович Богданов; премиальный, должник.",  "order\_details\_desc": "объемный, ценный.",  "order\_due\_date": "2024-04-08",  "order\_fact\_completion\_date": "2024-04-27",  "order\_parts": [  "дисплей",  "корпус"  ],  "repair\_types": [  "пайка"  ],  "order\_price": 7472.5287,  "order\_master\_id": 871908  }  },  {  "index": "order",  "id": 683490,  "body": {  "order\_id": 683490,  "order\_date": "2024-01-22",  "order\_customer\_id": 269146,  "order\_customer\_desc": "Имя: Алина Оскаровна Орехова; постоянный, должник.",  "order\_details\_desc": "ценный, тяжелый.",  "order\_due\_date": "2024-04-15",  "order\_fact\_completion\_date": "2024-04-20",  "order\_parts": [  "корпус",  "корпус"  ],  "repair\_types": [  "анализ неисправности",  "анализ неисправности"  ],  "order\_price": 2959.11938941,  "order\_master\_id": 300714  }  },  {  "index": "order",  "id": 243483,  "body": {  "order\_id": 243483,  "order\_date": "2024-02-01",  "order\_customer\_id": 408481,  "order\_customer\_desc": "Имя: Филатов Карл Алексеевич; должник, премиальный.",  "order\_details\_desc": "объемный, тяжелый.",  "order\_due\_date": "2024-04-05",  "order\_fact\_completion\_date": "2024-04-27",  "order\_parts": [  "дисплей",  "корпус",  "дисплей"  ],  "repair\_types": [  "анализ неисправности",  "пайка"  ],  "order\_price": 2323.37152,  "order\_master\_id": 736774  }  },  ...  ] |

**3. Результат запроса №1 к ElasticSearch**

|  |
| --- |
| [{'key\_as\_string': '2023-12-01',  'key': 1701388800000,  'doc\_count': 13,  'over\_parts': {'doc\_count\_error\_upper\_bound': 0,  'sum\_other\_doc\_count': 0,  'buckets': [{'key': 'usb-разъем', 'doc\_count': 5},  {'key': 'дисплей', 'doc\_count': 4},  {'key': 'корпус', 'doc\_count': 4},  {'key': 'аккумулятор', 'doc\_count': 3}]}},  {'key\_as\_string': '2024-01-01',  'key': 1704067200000,  'doc\_count': 48,  'over\_parts': {'doc\_count\_error\_upper\_bound': 0,  'sum\_other\_doc\_count': 0,  'buckets': [{'key': 'usb-разъем', 'doc\_count': 23},  {'key': 'дисплей', 'doc\_count': 22},  {'key': 'корпус', 'doc\_count': 18},  {'key': 'аккумулятор', 'doc\_count': 16}]}},  {'key\_as\_string': '2024-02-01',  'key': 1706745600000,  'doc\_count': 37,  'over\_parts': {'doc\_count\_error\_upper\_bound': 0,  'sum\_other\_doc\_count': 0,  'buckets': [{'key': 'дисплей', 'doc\_count': 19},  {'key': 'usb-разъем', 'doc\_count': 14},  {'key': 'аккумулятор', 'doc\_count': 14},  {'key': 'корпус', 'doc\_count': 14}]}},  {'key\_as\_string': '2024-03-01',  'key': 1709251200000,  'doc\_count': 2,  'over\_parts': {'doc\_count\_error\_upper\_bound': 0,  'sum\_other\_doc\_count': 0,  'buckets': [{'key': 'usb-разъем', 'doc\_count': 1},  {'key': 'аккумулятор', 'doc\_count': 1},  {'key': 'корпус', 'doc\_count': 1}]}}] |

**4. Результат запроса №2 к ElasticSearch**

|  |
| --- |
| {'total': {'value': 8, 'relation': 'eq'},  'max\_score': 0.8435577,  'hits': [{'\_index': 'master',  '\_type': '\_doc',  '\_id': '22653',  '\_score': 0.8435577,  '\_source': {'master\_id': 22653,  'master\_desc': 'Акулина Рудольфовна Никитина, Стаж Работы: 15 Л./Г..',  'master\_feedbacks': ['ворчливый, аккуратный.',  'ворчливый, медлительный.']}},  {'\_index': 'master',  '\_type': '\_doc',  '\_id': '300714',  '\_score': 0.8435577,  '\_source': {'master\_id': 300714,  'master\_desc': 'Вероника Петровна Силина, Стаж Работы: 12 Л./Г..',  'master\_feedbacks': ['ворчливый, медлительный.',  'ворчливый, аккуратный.']}},  {'\_index': 'master',  '\_type': '\_doc',  '\_id': '405064',  '\_score': 0.8435577,  '\_source': {'master\_id': 405064,  'master\_desc': 'Хохлов Олег Харлампьевич, Стаж Работы: 3 Л./Г..',  'master\_feedbacks': ['ворчливый, медлительный.',  'ворчливый, аккуратный.',  'ворчливый, аккуратный.']}},  {'\_index': 'master',  '\_type': '\_doc',  '\_id': '257458',  '\_score': 0.8435577,  '\_source': {'master\_id': 257458,  'master\_desc': 'Шестакова Наина Владиславовна, Стаж Работы: 9 Л./Г..',  'master\_feedbacks': ['аккуратный, ворчливый.', 'ворчливый, аккуратный.']}},  {'\_index': 'master',  '\_type': '\_doc',  '\_id': '814777',  '\_score': 0.8435577,  '\_source': {'master\_id': 814777,  'master\_desc': 'Шарова Жанна Аркадьевна, Стаж Работы: 1 Л./Г..',  'master\_feedbacks': ['ворчливый, медлительный.',  'медлительный, аккуратный.',  'ворчливый, медлительный.']}},  {'\_index': 'master',  '\_type': '\_doc',  '\_id': '847121',  '\_score': 0.8435577,  '\_source': {'master\_id': 847121,  'master\_desc': 'Жуков Ладимир Всеволодович, Стаж Работы: 12 Л./Г..',  'master\_feedbacks': ['медлительный, ворчливый.',  'аккуратный, медлительный.']}},  {'\_index': 'master',  '\_type': '\_doc',  '\_id': '163543',  '\_score': 0.8435577,  '\_source': {'master\_id': 163543,  'master\_desc': 'Максимильян Виленович Гурьев, Стаж Работы: 11 Л./Г..',  'master\_feedbacks': ['аккуратный, медлительный.']}},  {'\_index': 'master',  '\_type': '\_doc',  '\_id': '785701',  '\_score': 0.8435577,  '\_source': {'master\_id': 785701,  'master\_desc': 'Панфилова Пелагея Игоревна, Стаж Работы: 8 Л./Г..',  'master\_feedbacks': ['медлительный, аккуратный.',  'медлительный, ворчливый.']}}]} |

# **ПРИЛОЖЕНИЕ Б**

Исходные коды программ

Листов 9

**1. Программа генерации JSON документов**

|  |
| --- |
| from faker import Faker  import json  import random  fake = Faker('ru\_RU') **# русскоязычные ФИО**  master\_id\_set = set() **# существующие id мастеров**  **# функция генерации записи о заказе**  def generate\_order\_data():  order\_id = fake.pyint(min\_value=1, max\_value=1000000)  order\_customer\_id = fake.pyint(min\_value=1, max\_value=1000000)  order\_customer\_name = fake.name() # ФИО  **# описание: ФИО + доп. информация**  smp = random.sample(["премиальный", "постоянный", "должник"], 2)  order\_customer\_desc = f'Имя: {order\_customer\_name}; {smp[0]}, {smp[1]}.'  smp = random.sample(["хрупкий", "тяжелый", "объемный", "ценный"], 2)  order\_details = f'{smp[0]}, {smp[1]}.'  **# генерация дат в заданном промежутке**  order\_date = fake.date\_between(start\_date="-70d",  end\_date="today")  order\_due\_date = fake.date\_between(start\_date="+30d", end\_date="+60d")  order\_completion\_date = fake.date\_between(start\_date="+30d", end\_date="+60d")  order\_parts = [fake.word(ext\_word\_list=['корпус', 'usb-разъем', 'дисплей', 'аккумулятор']) for \_ in range(fake.pyint(min\_value=1, max\_value=3))]  repair\_types = [fake.word(ext\_word\_list=['замена компонентов', 'пайка', 'анализ неисправности', 'мелкие детали']) for \_ in range(fake.pyint(min\_value=1, max\_value=2))]  order\_price = fake.pyfloat(min\_value=1000, max\_value=10000)  **# исполнитель - из множества существующих**  order\_master\_id = random.choice(list(master\_id\_set))  return {  "index": "order",  "id": order\_id,  "body": {  "order\_id": order\_id, **#id\_заказа**  "order\_date": order\_date.strftime("%Y-%m-%d"), **#дата\_заказа**  "order\_customer\_id": order\_customer\_id, **#id\_заказчика**  "order\_customer\_desc": order\_customer\_desc, **#сведения\_о\_заказчике**  "order\_details\_desc": order\_details, **#данные\_о\_заказе**  "order\_due\_date": order\_due\_date.strftime("%Y-%m-%d"), **#срок\_выполнения\_заказа**  "order\_fact\_completion\_date": order\_completion\_date.strftime("%Y-%m-%d"), **#фактическая\_дата\_выполнения**  "order\_parts": order\_parts, **#запчасть**  "repair\_types": repair\_types, **#ремонт**  "order\_price": order\_price, **#стоимость**  "order\_master\_id": order\_master\_id **#id\_мастера**  }  }  **# генерация записи о мастере**  def generate\_master\_data():  master\_id = fake.pyint(min\_value=1, max\_value=1000000)  **# пополнение множества существующих мастеров**  master\_id\_set.add(master\_id)  master\_name = fake.name()  master\_feedbacks = []    for \_ in range(fake.pyint(min\_value=1, max\_value=3)):  smp = random.sample(["аккуратный", "медлительный", "ворчливый"], 2)  master\_feedback = f'{smp[0]}, {smp[1]}.'  master\_feedbacks.append(master\_feedback)    master\_xp = f'Стаж работы: {fake.random\_int(min=1, max=15)} л./г.'  if fake.random\_int(min=1, max=5) == 2:  master\_edu = 'Тип образования: высшее'  else :  master\_edu = 'Тип образования: среднее'  return {  "index": "master",  "id": master\_id,  "body": {  "master\_id": master\_id,  "master\_desc": f'{master\_name}, {fake.sentence(ext\_word\_list=[master\_xp, master\_edu], nb\_words=1, variable\_nb\_words=True)}', **#сведения\_о\_мастере**  "master\_feedbacks": master\_feedbacks #отзывы\_о\_работе  }  }  **# сгенерировать 20 и 100 записей, записать в файлы**  masters = [generate\_master\_data() for \_ in range(20)]  orders = [generate\_order\_data() for \_ in range(100)]  with open('../masters.json','w') as f:  json.dump(masters,f)  with open('../orders.json','w') as f:  json.dump(orders,f) |

**2. Программа заполнения ES**

|  |
| --- |
| import json  from elasticsearch import Elasticsearch  **# подключение**  client = Elasticsearch([{"host": "127.0.0.1", "scheme": "http", "port": 9200}])  **# настройки анализатора**  analyzer\_settings = {  "settings": {  "analysis": {  "filter": {  "ru\_stop": {  "type": "stop",  "stopwords": "\_russian\_"  },  "snow\_ru\_stemmer": {  "type": "snowball",  "language": "russian"  }  },  "analyzer": {  "custom\_analyzer": {  "type": "custom",  "tokenizer": "standard",  "filter": [  "lowercase",  "ru\_stop",  "snow\_ru\_stemmer"  ]  }  }  }  }  }  **# маппинг заказов**  mappings\_order = {  "mappings" : {  "properties" : {  "order\_customer\_desc" : {  "type" : "text",  "analyzer": "custom\_analyzer"  },  "order\_customer\_id" : {  "type" : "long"  },  "order\_date" : {  "type" : "date"  },  "order\_details\_desc" : {  "type" : "text",  "analyzer": "custom\_analyzer"  },  "order\_due\_date" : {  "type" : "date"  },  "order\_fact\_completion\_date" : {  "type" : "date"  },  "order\_id" : {  "type" : "long"  },  "order\_master\_id" : {  "type" : "long"  },  "order\_parts" : {  "type" : "text",  "fielddata" : True,  "fields" : {  "keyword" : {  "type" : "keyword"  }  }  ,"analyzer": "custom\_analyzer"  },  "order\_price" : {  "type" : "float"  },  "repair\_types" : {  "type" : "text",  "fields" : {  "keyword" : {  "type" : "keyword"  }  }  ,"analyzer": "custom\_analyzer"  }  }  }  }  **# (пере)создание индекса**  indexName = "order"  if client.indices.exists(index=indexName):  client.indices.delete(index=indexName)  client.indices.create(index=indexName,  body={\*\*analyzer\_settings  , \*\*mappings\_order  }  )  **# заполнение индекса**  with open('../orders.json', 'r') as f:  dataStore = json.load(f)  for data in dataStore:  try:  client.index(  index=data["index"],  # doc\_type=data["doc\_type"],  id=data["id"],  body=data["body"]  )  except Exception as e:  print(e, end="")  print("index 'order' data inserted")  **# маппинг мастеров**  mappings\_master = {  "mappings" : {  "properties" : {  "master\_desc" : {  "type" : "text",  "analyzer": "custom\_analyzer"  },  "master\_feedbacks" : {  "type" : "text",  "fields" : {  "keyword" : {  "type" : "keyword"  }  }  ,"analyzer": "custom\_analyzer"  },  "master\_id" : {  "type" : "long"  }  }  }  }  **# (пере)создание индекса**  indexName = "master"if client.indices.exists(index=indexName):  client.indices.delete(index=indexName)  client.indices.create(index=indexName,  body={\*\*analyzer\_settings  , \*\*mappings\_master  }  )  **# заполнение индекса**  with open('../masters.json', 'r') as f:  dataStore = json.load(f)  for data in dataStore:  try:  client.index(  index=data["index"],  id=data["id"],  body=data["body"]  )  except Exception as e:  print(e, end="")  print("index 'master' data inserted") |

**3. Программа заполнения Neo4j**

|  |
| --- |
| import json  from elasticsearch import Elasticsearch  from py2neo import Graph, Node, Relationship  **# подключение к ES**  client = Elasticsearch([{"host": "127.0.0.1", "scheme": "http", "port": 9200}])  **# подключение к Neo4j**  graph\_db = Graph("bolt://localhost:7687", user='neo4j', password='my\_password')  searchBody = {  "size": 9999,  "\_source": True,  "query": {  "match\_all": {}  }  }  **# получение всех мастеров**  result = client.search(index="master", body=searchBody)  masters = result['hits']['hits']  **# очистка Neo4j**  graph\_db.run("MATCH (n) OPTIONAL MATCH (n)-[r]-() DELETE n,r;")  i = 0  **# для каждого мастера: ...**  for master in masters:  try:  print(f'{i})', end='')  **# создать узел**  MasterNode = Node("Master",master\_id=master['\_source']['master\_id'], master\_desc=master['\_source']['master\_desc'])  graph\_db.create(MasterNode)  **# найти соответствующие заказы**  searchBody = {  "size": 9999,  "\_source": True,  "query": {  "term": {  "order\_master\_id": master['\_source']['master\_id']  }  }  }  result = client.search(index="order", body=searchBody)  orders = result['hits']['hits']  **# создать узлы заказов и связи с ними**  for order in orders:  OrderNode = Node("Order",order\_id=order['\_source']['order\_id'], order\_date=order['\_source']['order\_date'], order\_customer\_desc=order['\_source']['order\_customer\_desc'], order\_price=order['\_source']['order\_price'])  graph\_db.create(OrderNode)  NodesRelationship = Relationship(MasterNode, "Executed", OrderNode, order\_due\_date=order['\_source']['order\_due\_date'], order\_fact\_completion\_date=order['\_source']['order\_fact\_completion\_date'])  graph\_db.create(NodesRelationship)  print('.', end='')  print("+")  i += 1  except Exception as e:  print(f'Exception: {e}')  continue  print("data written successfully") |

**4. Программа формирования csv-файлов в HDFS**

|  |
| --- |
| from pyspark.sql import SparkSession  from elasticsearch import Elasticsearch  from pyspark.sql.types import \*  import findspark  findspark.init()  **# подключение к ES и Spark**  client = Elasticsearch("http://localhost:9200")  sparkSession = SparkSession.builder.appName("csv4").getOrCreate()  **# получение всех мастеров и заказов из ES**  searchBody = {  "size": 9999,  "\_source": True,  "query": {  "match\_all": {}  }  }  result = client.search(index="master", body=searchBody)  masters = result['hits']['hits']  result = client.search(index="order", body=searchBody)  orders = result['hits']['hits']  **# описание схем**  MasterSchema = StructType([  StructField("master\_id", IntegerType(), False),  StructField("master\_desc", StringType(), False),  StructField("master\_feedbacks", StringType(), False)  ])  CustomerSchema = StructType([  StructField("order\_customer\_id", IntegerType(), False),  StructField("order\_customer\_desc", StringType(), False)  ])  OrderSchema = StructType([  StructField("order\_id", IntegerType(), False),  StructField("order\_date", StringType(), False),  StructField("order\_customer\_id", IntegerType(), False),  StructField("order\_details\_desc", StringType(), False),  StructField("order\_due\_date", StringType(), False),  StructField("order\_fact\_completion\_date", StringType(), False),  StructField("order\_parts", StringType(), False),  StructField("repair\_types", StringType(), False),  StructField("order\_price", FloatType(), False),  StructField("order\_master\_id", IntegerType(), False)  ])  **# заполнение массивов**  MasterTable = []  CustomerTable = []  OrderTable = []  CustomerSet = set()  for master in masters:  MasterTable.append((  master['\_source']['master\_id'],  master['\_source']['master\_desc'],  str(master['\_source']['master\_feedbacks'])  ))  for order in orders:  OrderTable.append((  order['\_source']['order\_id'],  order['\_source']['order\_date'],  order['\_source']['order\_customer\_id'],  order['\_source']['order\_details\_desc'],  order['\_source']['order\_due\_date'],  order['\_source']['order\_fact\_completion\_date'],  str(order['\_source']['order\_parts']),  str(order['\_source']['repair\_types']),  order['\_source']['order\_price'],  order['\_source']['order\_master\_id']  ))  if not order['\_source']['order\_customer\_id'] in CustomerSet:  CustomerSet.add(order['\_source']['order\_customer\_id'])  CustomerTable.append((  order['\_source']['order\_customer\_id'],  order['\_source']['order\_customer\_desc']  ))  **# преоброазование в dataframe**  MasterDF = sparkSession.createDataFrame(MasterTable, MasterSchema)  OrderDF = sparkSession.createDataFrame(OrderTable, OrderSchema)  CustomerDF = sparkSession.createDataFrame(CustomerTable,CustomerSchema)  **# запись в HDFS**  MasterDF.write.csv(path='hdfs://0.0.0.0:9000/master.csv',mode='overwrite', header=True)  OrderDF.write.csv(path='hdfs://0.0.0.0:9000/order.csv',mode='overwrite', header=True)  CustomerDF.write.csv(path='hdfs://0.0.0.0:9000/customer.csv',mode='overwrite', header=True)  print('data is written successfully')  sparkSession.stop() **# закрытие сессии** |

**5.  Программа реализации запроса для работы монитора**

|  |
| --- |
| import findspark  from pyspark.sql import SparkSession  findspark.init()  **# подключение к Spark**  sparkSession = SparkSession.builder.appName("csv4").getOrCreate()  **# считывание csv и регистрация временных таблиц**  data = sparkSession.read.load("hdfs://localhost:9000/master.csv", format="csv",sep=",", inferSchema="true", header="true")  data.registerTempTable("master")  data = sparkSession.read.load("hdfs://localhost:9000/order.csv", format="csv",sep=",", inferSchema="true", header="true")  data.registerTempTable("order")  data = sparkSession.read.load("hdfs://localhost:9000/customer.csv", format="csv",sep=",", inferSchema="true", header="true")  data.registerTempTable("customer")  **# исполнение запроса и вывод результата**  result = sparkSession.sql("""select m.master\_id, m.master\_desc, o.order\_id, o.order\_date, o.order\_due\_date, o.order\_fact\_completion\_date, c.order\_customer\_id, c.order\_customer\_desc  from master m JOIN order o ON o.order\_master\_id = m.master\_id LEFT JOIN customer c ON o.order\_customer\_id = c.order\_customer\_id  where o.order\_fact\_completion\_date > o.order\_due\_date  """)  result.show()  Input()  sparkSession.stop() **# закрытие сессии** |

# **ПРИЛОЖЕНИЕ В**

Гафический материал

Листов 11

