



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

ЛЕКЦИОННЫЕ МАТЕРИАЛЫ

Технологии хранения в системах кибербезопасности

(наименование дисциплины (модуля) в соответствии с учебным планом)

Уровень

бакалавриат

(бакалавриат, магистратура, специалитет)

Форма обучения

очная

(очная, очно-заочная, заочная)

Направление(-я)
подготовки

10.05.04 Информационно-аналитические системы безопасности

(код(-ы) и наименование(-я))

Институт

Кибербезопасности и цифровых технологий (ИКБ)

(полное и краткое наименование)

Кафедра

КБ-2 «Прикладные информационные технологии»

(полное и краткое наименование кафедры, реализующей дисциплину (модуль))

Лектор

к.т.н., Селин Андрей Александрович

(сокращенно – ученая степень, ученое звание; полностью – ФИО)

Используются в данной редакции с учебного года

2024/2025

(учебный год цифрами)

Проверено и согласовано «___» _____ 2024 г.

А.А. Бакаев

*(подпись директора Института/Филиала
с расшифровкой)*

Москва 2024 г.



Технологии хранения в системах кибербезопасности

2024 год



Лекция 8. Elasticsearch

Учебные вопросы лекции:

1. ES
2. Типы поиска в ES

Введение

ElasticSearch – NoSQL БД/поисковая система с открытым исходным кодом, предназначенная для полнотекстового поиска. Она позволяет хранить, анализировать и получать большие объемы данных в режиме реального времени.

Для анализа и поиска ElasticSearch использует библиотеку Apache Lucene. Написана она на языке Java и доступна для многих платформ. Все неструктурированные данные хранятся в формате JSON. Для работы с данными у Elastic Search есть специальное REST API.

Основные задачи, решаемые ElasticSearch:

- полнотекстовый поиск;
- поиск по параметрам;
- Агрегация данных для статистики и их последующая визуализация;
- Генерация вариантов для автозаполнения.



ES

Эквивалентные термины в Elasticsearch.

Например тип для лекций:

```
{  
  "articleid": 1,  
  "name": "Elasticsearch"  
}
```

Тип для учебных вопросов:

```
{  
  "commentid": "WetKRvTNWfhuxCa_lTcb",  
  "articleid": 1,  
  "topic": "ES"  
}
```


При желании можно определить отношения между различными **типами**. Например, отношение между родителями и дочерними элементами может быть определено между лекциями и комментариями. Лекция (родитель) может иметь один или несколько учебных вопросов (детей).

ES

Кроме того Elasticsearch поддерживает хранение вложенных объектов:

```
{
  "id": 2,
  "name": "Мария",
  "age": 20,
  "gender": "Ж",
  "email": "enkova@mirea.ru",
  "address": {
    "street": "Проспект Вернадского",
    "city": "Москва",
    "state": "Россия",
    "zip": 98765
  }
}
```

```
{
  "id": 1,
  "name": "Василий",
  "age": 21,
  "gender": "М",
  "email": "zaicev@mirea.ru"
},
{
  "id": 2,
  "name": "Мария",
  "age": 20,
  "gender": "Ж",
  "email": "enkova@mirea.ru"
}
```



id	name	age	gender	email
1	Василий	21	М	zaicev@mirea.ru
2	Мария	20	Ж	enkova@mirea.ru

ES

Elasticsearch	SQL	MongoDB
Index		
М		

Apache Lucene – это высокопроизводительная полнофункциональная библиотека поисковой системы, полностью написанная на Java. Технология, подходящая практически для любого приложения, требующего структурированного поиска, полнотекстового поиска, поиска ближайших соседей по векторам большой размерности и др.

Индекс

Индекс

похож на БД в смысле хранения данных. Индекс очень похож с индексом БД, как в реляционных БД. Типы данных (таблиц). Имя индекса должно быть уникальным и состоять из строчных букв.

Elasticsearch использует индексы *Lucene* для хранения данных и поиска.



Shard в **Elasticsearch** – это логическая единица хранения данных на уровне базы, которая является отдельным экземпляром Lucene.

один или более **шардов**, их совокупность и является хранилищем.

ES

Для того, чтобы обращаться к распределенной системе шардов, нам необходимо иметь некий координирующий узел, именно он будет принимать запросы и давать задания на запись или получение данных. То есть помимо хранения данных мы выделяем еще один вариант поведения программы — координирование.

Узел. Узловая машина является частью сервера и называется отдельной машиной. Она хранит данные и предоставляет возможности индексации и поиска, а также другие узлы кластера. Благодаря концепции горизонтального масштабирования возможно виртуально добавить бесконечное количество узлов в кластер ES, чтобы дать ему гораздо больше возможностей и, в том числе, возможностей индексирования.

Есть два вида узлов — CRUD-узлы и координирующие узлы. Назовем их **data node** и **coordinating node**. Несколько машин, объединенных в сеть напоминают кластер.

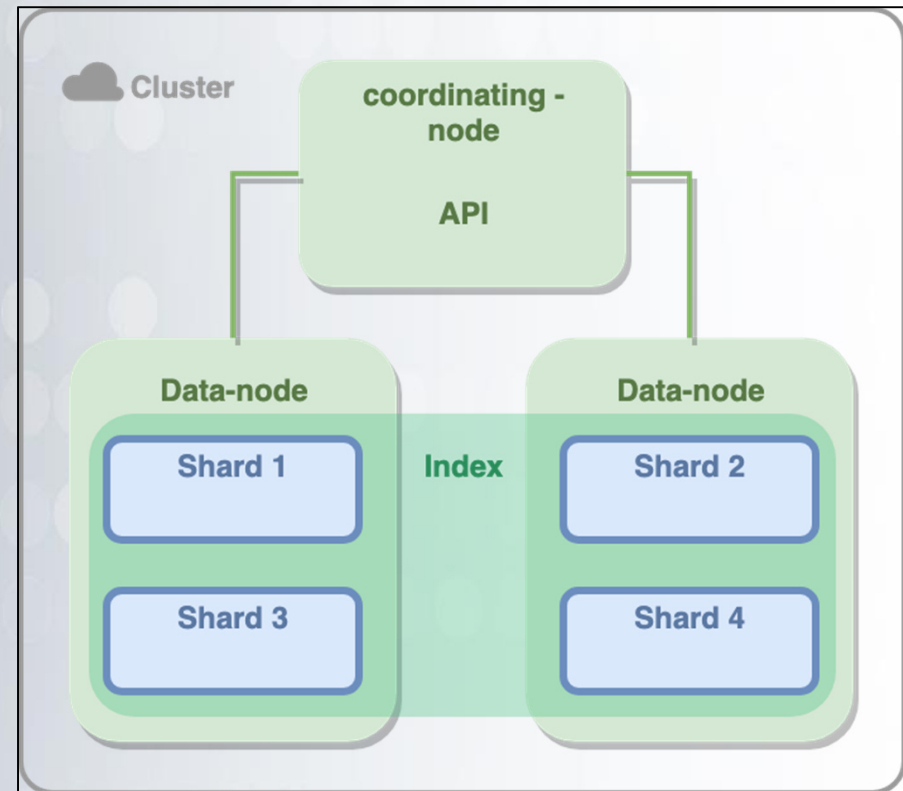
ES

Кластер – это набор серверных машин (узлов), на которых хранятся данные. Данные разделены между несколькими узлами, чтобы их можно было реплицировать, и единственная точка отказа (SPoF) не возникла с ES-сервером. Имя кластера по умолчанию: Elasticsearch. Каждый узел в кластере подключается к кластеру с помощью URL-адреса и имени кластера, поэтому важно, чтобы это имя было четким и понятным.

Каждый запущенный экземпляр Elasticsearch является отдельным узлом (node). Cluster – это совокупность определенных нод. Когда вы запускаете один экземпляр, ваш кластер будет состоять из одной ноды.

Для того чтобы объединить узлы в кластер, они должны соответствовать ряду требований:

- Ноды должны иметь одинаковую версию.
- Имя кластера `cluster.name` в конфигурации должно быть одинаковым.



ES

Очевидно, что data-ноды будут часто обращаться к диску и использовать значительные объемы памяти в процессе работы. Не все данные будут запрашиваться одинаково часто. Данные постепенно "остывают" по мере снижения запросов (жизненный цикл хранения данных). Популярные публикации нужно хранить там, откуда их можно быстро достать, а забытые «анекдоты 90-х» можно положить подальше.

Начиная с версии 6.7 Elasticsearch предлагает механизм управления жизненным циклом. Для этого доступны три типа нод:

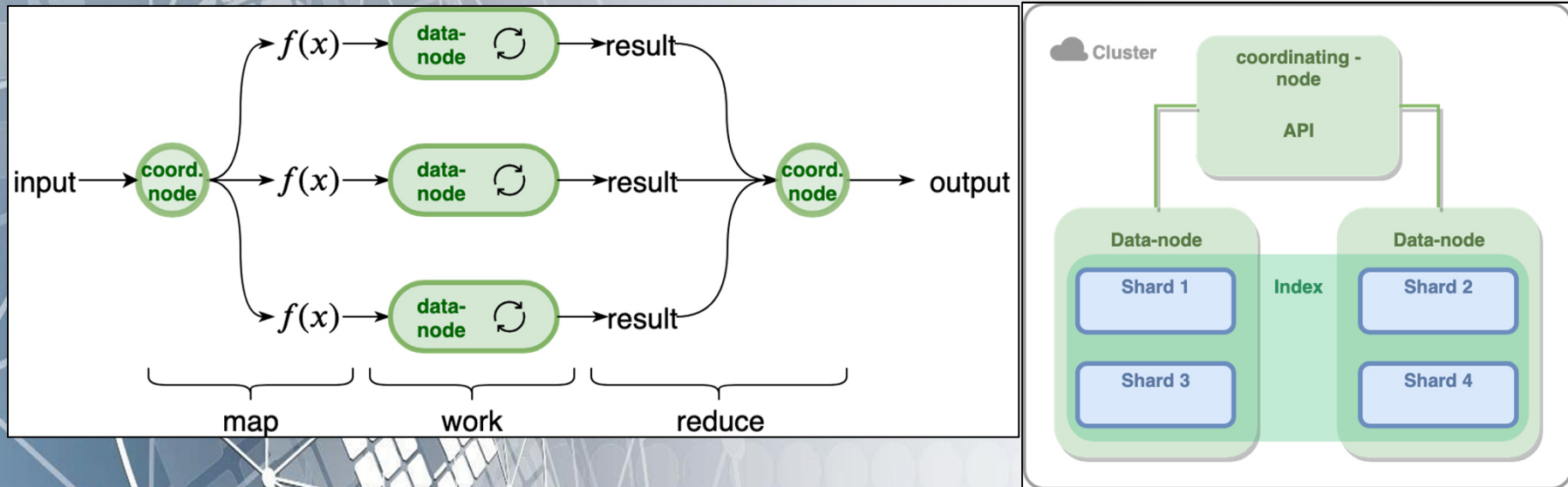
- hot,
- warm,
- cold.

Существует рекомендация по выбору аппаратных конфигураций для каждого из типов. Например hot-ноды должны иметь быстрые SSD, для warm и cold достаточно HDD-диска. Оптимальные соотношения память/диск будут следующими:

hot — 1:30
warm — 1:100
cold — 1:500

ES

Важнейшим аспектом в использовании распределенных систем является параллельное выполнение задач.



Именно такой механизм помогает выполнять операции с шардами.

Координирующий узел получит запрос, предварительно переформулирует его для внутрикластерного взаимодействия и выполнит запросы к **worker-нодам** (в данном случае к **data-нодам**).

Следовательно, **coordinating-ноды** должны иметь достаточный ресурс памяти, ЦП и быструю сеть, но при этом могут иметь скромный диск, ведь не осуществляют хранения данных.

ES

Управление кластером

Предположим возможность **coordinating-нодам** управления состоянием кластера. Один узел примет решение о перемещении шарда на одну **data-ноду**, а второй о перемещении того же на другую. Список возможных общекластерных действий может быть довольно широким, а список возможных конфликтов еще шире.

Такие важные решения должен принимать один центральный узел. Для каждого типа действий необходимо выделять отдельную роль, чтобы избежать потерь производительности на ноде. И "главный в кластере" звучит как отдельная ответственность. Такие ноды называются **master-node**. Активный мастер всегда должен быть один, он будет управлять топологией кластера: создавать новый индекс, выделять и распределять шарды, перемещать их и объединять в случае необходимости. Мастер всегда знает все о состоянии кластера.

В кластере **Elasticsearch** обязательно должен быть как минимум один узел, отвечающий требованиям master node. Для этого в конфигурации ноды необходимо установить значение

node.master: true.

Master-ноды отвечают за важные, но довольно легкие общекластерные действия. Это означает, что они требуют большого ресурса и высокой стабильности от физической ноды. В кластерах от 10 нод необходимо всегда выделять отдельные **only-master узлы**.

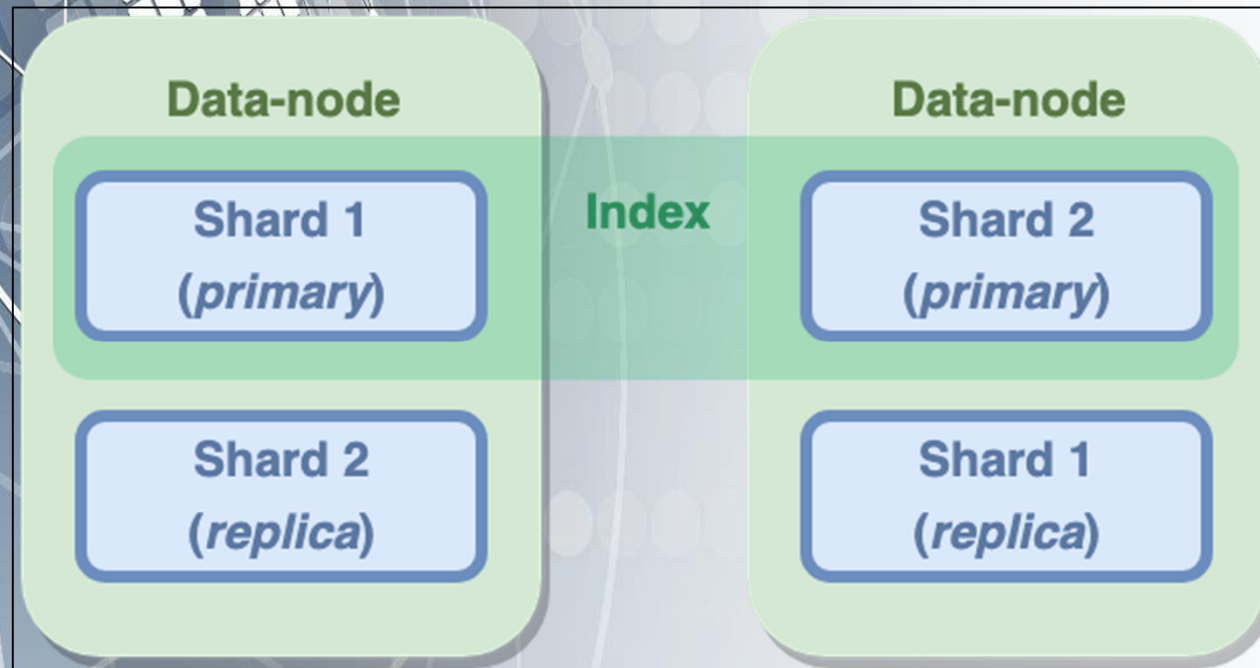
ES

Репликация данных

Чтобы жестко установить количество реплик индекса используется параметр `number_of_replicas`. Так же мы можем изменить это значение в рантайме выполнив запрос:

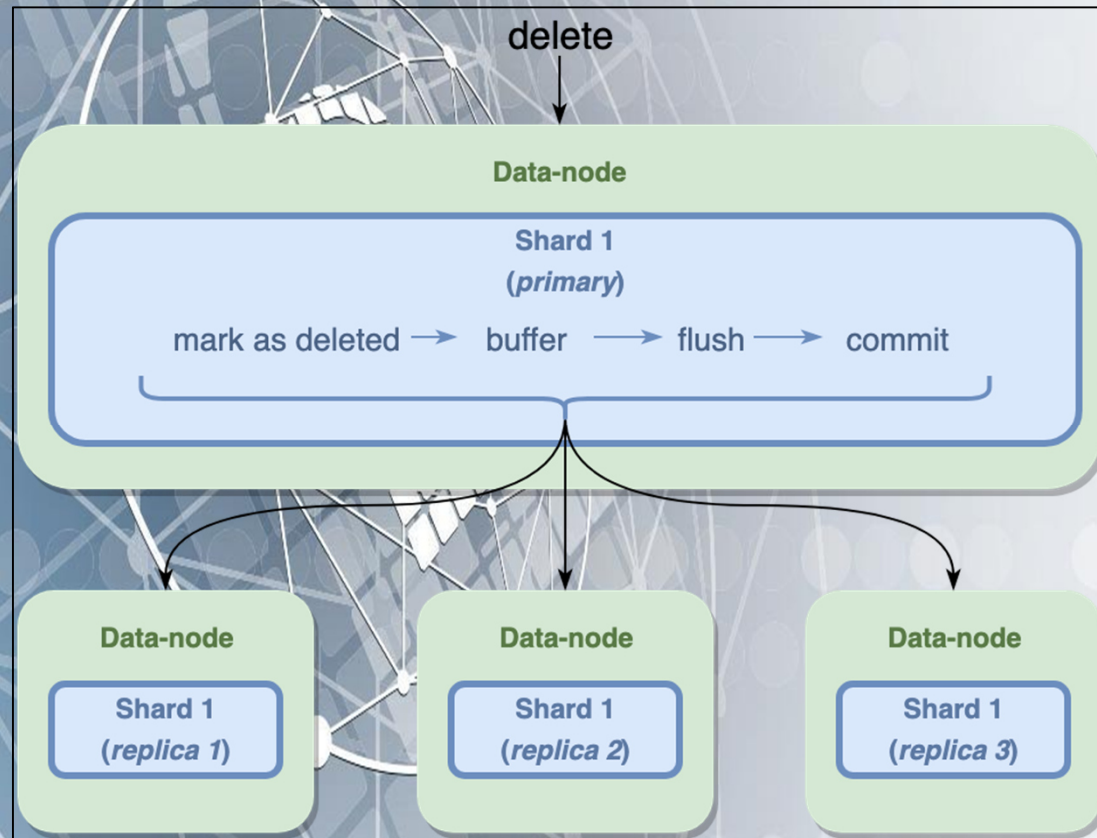
```
PUT /_settings {  
  "index": {  
    "number_of_replicas": someVal  
  }  
}
```

Но эти узлы должны хранить только шарды реплик.



ES

С учетом реплик запись данных будет происходить в два этапа, в первом запись затронет только первичный шард и только после того, как произойдет операция **flush** слияния изменений и операция **commit** фиксации в индексе Lucene, будет отправлен внутренний запрос на изменение всех реплик.



Для мониторинга состояния кластера в Elasticsearch существует `cluster health status`. Статус может иметь три значения:

- green** – все хорошо;
- yellow** – есть утраченные шарды, кластер полностью работоспособен, но работает на репликах;
- red** – есть утраченные шарды, кластер неработоспособен или часть данных недоступна.

Для максимальной стабильности кластера необходимо, чтобы количество дата-нод было больше или равно количеству реплик.

Отказоустойчивость

Теперь данные будут доступны даже в случае сбоя одного из хранящих узлов. Но что будет если кластер потеряет мастера? Потеря единственного мастера равноценна потере кластера. Вариант выхода – поднять несколько мастеров.

Если есть, например, два управляющих узла, как понять, какой из них в данный момент должен управлять кластером? Как они смогут договориться о своих решениях?

В каждый момент времени должен быть только один управляющий кластером узел. То есть при потере мастера его место должен занять один из кандидатов.

В Elasticsearch настройка `node.master: true` не будет означать, что данный узел является мастером, это всего лишь скажет о том, что он может быть выбран в качестве главного узла. По умолчанию настройки будут установлены следующим образом:

- `node.master: true`
- `node.data: true`

ES

Пример. Главный управляющий узел стал недоступен для кластера, кластер берет первого кандидата и устанавливает его на вакантное место. Спустя определенное время первый мастер возвращается в кластер и ничего не знает о том, что его место уже занято. Мастер-ноды являются своего рода его *мозгом*, и теперь мозг кластера становится разделен. Это классическая проблема распределенных систем и она так и называется *split-brain problem*.



Как только кластер теряет управляющий узел, должен быть запущен процесс голосования. Важно определить какой из кандидатов больше всего подходит на роль главного узла. Такой кандидат должен обладать самой актуальной информацией о кластере.

Сравнив номера версий мы можем определить наиболее подходящих кандидатов на роль мастера. Теперь если отправшая мастер-нода вернется в кластер, то процесс голосования запустится снова и будет выбран единственный управляющий узел.

ES

Теперь важно понять когда можно считать, что голосование прошло успешно? Если проголосовали все участники? Или половина? Или другое любое другое количество? Решение этой проблемы заключается в определении *кворума*.

Это умное название для контрольного количества голосующих. Очевидно, что такое важное решение как выбор мастера должно приниматься на основе большинства, то есть 50%+один голос. Справедливо, надежно. Это значение и станет кворумом.

Таким образом, количество кандидатов на мастера должно быть нечетным и не меньше трех. Рекомендуется использовать простую формулу для расчета оптимально количества таких нод:

$$\text{КОЛИЧЕСТВО_КАНДИДАТОВ} = \text{ОБЩЕЕ_КОЛИЧЕСТВО_НОД} / 2 + 1$$

Решения для любых общекластерных действий принимаются путем голосования, и вся необходимая для голосования информация содержится в *конфигурации голосования*. Право голоса определяет еще одну роль, ведь право голоса не означает, что узел может быть кандидатом.

В Elasticsearch узлы, которые могут участвовать в голосовании можно определить в конфигурации: **node.voting_only: true**.

ES

Транспорт

Пришло время поговорить о том, как общаться с кластером из внешних систем, и как будут общаться узлы внутри кластера. Есть ряд плюсов и минусов использования и *традиционных*, и специальных протоколов. Для краткого сравнения существует таблица.

Протокол	Достоинства	Недостатки
HTTP	Низкий порог вхождения, в сравнении с нативным протоколом. Для использования нужен только HTTP клиент. HTTP API никогда не ломает совместимость, при обновлении версии ES, приложение продолжит работать так же. Возможно проксировать и использовать балансировщики нагрузки. JSON.	Клиент не знает топологию кластера, поэтому может потребовать большее количество запросов для получения данных.
ES Native	Лучший выбор для ОЧЕНЬ больших данных. Если необходимо выполнить большое количество операций с индексом, нативный протокол значительно ускорит.	Используется под JVM. Использование влечет жесткую связность с ES. Обновления требуют перекомпиляции и повторного развертывания пользовательских клиентов. Возможны обновления ломающие совместимость.

Типы поиска в ES

Elasticsearch позволяет осуществлять поиск в режиме, близком к реальному времени с гибкостью, которую он предоставляет с типом индексируемых и просматриваемых данных.

Структурированный поиск: этот тип поиска выполняется по данным, имеющим predetermined формат, например даты, время и числа. С predetermined форматом появляется гибкость выполнения общих операций, таких как сравнение значений в диапазоне дат. Однако, **текстовые данные тоже могут быть структурированы**. Это возможно, если поле имеет фиксированное количество значений. Например, имя БД может быть:

MySQL,
MongoDB,
Elasticsearch,
и т. д.

При структурированном поиске ответ на запросы, которые мы выполняем, будет «да» либо «нет».

Типы поиска в ES

Полнотекстовый поиск: этот тип поиска зависит от двух важных факторов: **актуальности** и **анализа**. С помощью **Relevance (актуальности)** определяется, насколько хорошо некоторые данные соответствуют запросу, ставя оценку для результирующих документов. Эта оценка выставляется самим ES.

Анализ относится к разбиению текста на нормализованные токены для создания инвертированного индекса.

Многополевой поиск: количество аналитических запросов к хранимым в ES данным постоянно растет, поэтому мы обычно не сталкиваемся с простыми запросами на сопоставление. Требования выросли для выполнения запросов, которые

Запрос на совпадение найдет все три документа при поиске *Удар по мячу*. Поиск по близости может сказать нам, как далеко эти два слова появляются в одной строке или абзаце, из-за чего они совпадают.

Сопоставление с приоритетом: запросы сегодня - это гораздо больше, чем просто определение того, содержат ли некоторые текстовые данные другую строку или нет. Речь идет об установлении связи между данными, для их оценки и сопоставления с контекстом, в котором сравниваются данные, к примеру:

- Мяч попал в Петю
- Петя ударил по мячу
- Петя купил новый мяч, который попал в окно Василия

Типы поиска в ES

Частичное сопоставление, как становится понятно из названия, позволяет нам запускать запросы с частичным соответствием.

Аналогичные запросы на основе SQL(*частичное соответствие*)

Where имя like «Пет%»

and фамилия like "%вано%»

and дата_рождения like "%10.200%»

В некоторых случаях необходимо выполнять только запросы частичного совпадения, даже если их можно рассматривать как строгие запросы.



СПАСИБО ЗА ВНИМАНИЕ!

