



УНИВЕРСИТЕТ ИТМО

Проектирование Бд

Лекция 10. ElasticSearch

Про что мы сегодня? (в прошлый-то раз SQL)

MongoDB	ElasticSearch
Документо-ориентированная БД	Поисковая БД (тоже документо-...)
Написана на C++	Написана на Java
Поддержка транзакций (через изоляцию снимков состояний)	Нет поддержки транзакций
Поддержка SQL-запросов на чтение	Собственный язык запросов, похожий на SQL

Сравнение понятий

PostgreSQL (any SQL)	MongoDB	ElasticSearch
Схема	Схема	Индекс
Таблица	Коллекция	Маппинг {Тип} (с ES 7)
Кортеж	BSON-объект	JSON-объект
Атрибут	Поле	Поле

ElasticSearch (далее – ES)

Elasticsearch – нереляционное хранилище документов, которое предоставляет пользователю REST API и позволяет работать с данными в формате JSON.

Строгая структурированность при этом не является обязательным требованием.

Из чего состоит ES?

Кластер

Нода 1

Индекс 1

Шард 1 (R)

Сегменты

Шард 2 (R)

Сегменты

Индекс 2

Шард 5 (R)

Сегменты

Шард 6 (R)

Сегменты

Нода 2

Индекс 1

Шард 1 (R)

Сегменты

Шард 2 (R)

Сегменты

Индекс 2

Шард 5 (P)

Сегменты

Шард 6 (R)

Сегменты

Чуть поподробнее про translog

- При индексировании – новый сегмент (сначала пишется в память, по достижении условий Lucene commit – на диск)
- Чтобы не потерять до коммита данные из памяти при сбое – дополнительно они дублируются в translog
- Translog можно настроить на запись на определённую Δt
- Данные, не переданные в инвертированный индекс, можно прочитать по идентификатору из translog

Что в ES такого необычного?

ES — это, фактически, просто надстройка. Каждый шард в ES — полноценный индекс Apache Lucene. И данные, на самом деле, хранит именно он.

В чём же ценность ES? Помимо чисто пользовательской (предоставление удобного API поиска) — полноценная поддержка основных принципов распределённых баз данных.

А кто такой этот Apache Lucene?

AL – одна из лучших реализаций полнотекстового поиска, за счёт применения инвертированного индекса обеспечивающая очень высокую производительность.

Библиотека с открытым кодом (да, вы можете написать свою альтернативу ES)

И что он умеет?

Инвертированный индекс

Die Hard

Die Hard 2: Die Harder

Die Hard with a Vengeance

Live Free or Die Hard

A Good Day to Die Hard

Слово	Документ
Die	1, 2, 3, 4, 5
Hard	1, 2, 3, 4, 5
2	2
Harder	2
Vengeance	3
Live	4
Free	4
Good	5
Day	5

И что он умеет?

Инвертированный позиционный

Die Hard

Die Hard 2: Die Harder

Die Hard with a Vengeance

Live Free or Die Hard

A Good Day to Die Hard

Слово	Документ
Die	1: 1, 2: 1, 2: 4, 3: 1, 4: 3, 5: 3
Hard	1: 2, 2: 2, 3: 2, 4: 3, 5: 4
2	2: 3
Harder	2: 4
Vengeance	3: 3
Live	4: 1
Free	4: 2
Good	5: 1
Day	5: 2

И что ещё он умеет?

- **Токенизация** (разделения текста в поток токенов по их границам – пробелы, знаки препинания и т.п.)
- **Морфологическая трансформация** (стемминг - сокращение до корня, в индексе хранится корневое слово)
Подстановка синонимов (помогает при поиске слов с одинаковым смыслом, сокращает индекс)
- **Фильтр стоп-слов** (например, артикли)

Что к этому добавляет ES?

Можно настраивать и подключать в индекс дополнительные пользовательские анализаторы

Возвращаемся к ES. Про шарды

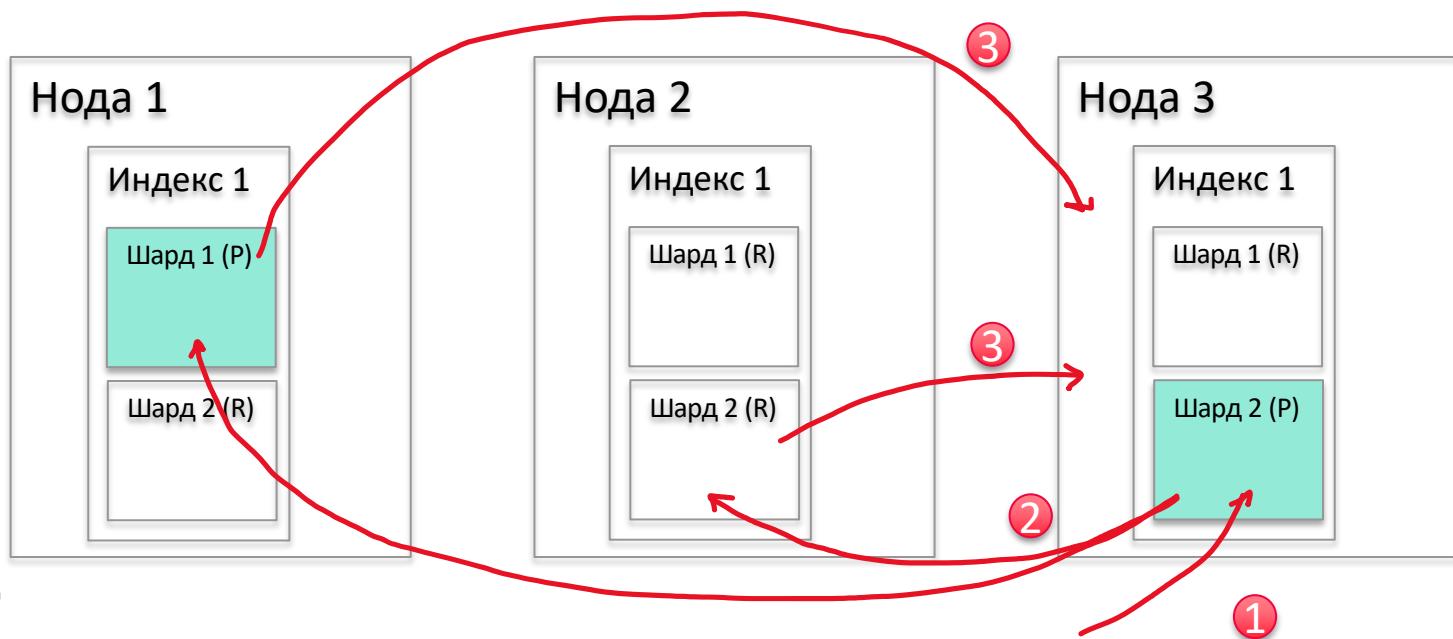
Когда мы индексируем («сохраняем в базу») – документ сохраняется на primary шард.

!! – количество primary шардов нельзя менять после создания базы, потому что оно используется при определении того, на какой именно шард будет помещён индексированный документ.

Маршрут индексации документа (запись)



Маршрут запроса получения документа



Про ES и CAP-теорему

Формально, ES ближе всего к CP

Если в структуре нагрузки преобладает чтение, то можно достичь AP-поведения системы, отказавшись от подтверждения с большого количества мастер-нод (но нам потребуется обеспечить доступность большого количества)

Напомним сравнение понятий

PostgreSQL (any SQL)	MongoDB	ElasticSearch
Схема	Схема	Индекс
Таблица	Коллекция	Маппинг {Тип} (с ES 7)
Кортеж	BSON-объект	JSON-объект
Атрибут	Поле	Поле

Схемы нет, а индекс есть. А что в нём?

```

4   "mappings": {
5     "properties": {
6       "age": {
7         "type": "long"
8       },
9       "famous_for": {
10         "type": "text",
11         "analyzer": "my_analyzer",
12         "search_analyzer": "my_stop_analyzer",
13         "search_quote_analyzer": "my_analyzer"
14     },
15     "name": {
16       "type": "text",
17       "fields": {
18         "keyword": {
19           "type": "keyword",
20           "ignore_above": 256
21         }
22       }
23     },
24     "team": {
25       "properties": {
26         "engine": {
27           "type": "text",
28           "fields": {
29             "keyword": {
30               "type": "keyword",
31               "ignore_above": 256
32             }
33           },
34         },
35         "name": {
36           "type": "text",
37           "fields": {
38             "keyword": {
39               "type": "keyword",
40               "ignore_above": 256
41             }
42           }
43         }
44       }
45     }
46   },
47 }
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
"settings": {
  "index": {
    "routing": {
      "allocation": {
        "include": {
          "_tier_preference": "data_content"
        }
      }
    },
    "number_of_shards": "1",
    "provided_name": "champions",
    "creation_date": "1713165889532",
    "analysis": {
      "filter": {
        "english_stop": {
          "type": "stop",
          "stopwords": "_english_"
        }
      },
      "analyzer": {
        "my_analyzer": {
          "filter": [
            "lowercase"
          ],
          "type": "custom",
          "tokenizer": "standard"
        },
        "my_stop_analyzer": {
          "filter": [
            "lowercase",
            "english_stop"
          ],
          "type": "custom",
          "tokenizer": "standard"
        }
      }
    },
    "number_of_replicas": "1",
    "uuid": "NSV6vtyzRR4q880CF0_5DnA",
    "version": {
      "created": "8503000"
    }
}

```

Небольшие лайфхаки с маппингом

!! - Можно исключать поля из индекса:

```
{ «mappings» : { «properties» : { «age» : { «type» : integer,  
«index» : false } } }
```

!! – Если документ рид-онли, ему можно отключить поле
_source (в котором хранится исходный JSON и которое
используется только в случае обновление)

```
{ «mappings» : { «_source» : { «enabled» : false} } }
```

Какие типы данных знает?

Text, Keyword – вместо String (начиная с версии 5.5 и далее)

Date

Numeric (long, integer, short, byte, double, float, half_float)

Boolean

Binary

Массивы, объекты, вложенные типы данных, геоданные

Хорошо, индекс есть. А JOIN-ы есть?

Есть три типа «объединений»:

- nested query
- has_parent
- has_child

!! – все объединённые документы должны храниться на
одном шарде

nested_query

```
"mapping": {  
    "properties": {  
        "team": {  
            "type": "nested",  
            "properties": {  
                "name": "text"  
            }  
        }  
    }  
}
```

- + отдельный объект
- + можно много уровней
- отдельный запрос
- переиндексация всего
- при поиске вернётся весь документ, а не только вложенный

has_parent/has_child

```
"mappings": {  
    "team": {  
        "properties": {  
            "name": {  
                "type": "string"  
            }  
        }  
    },  
    "driver": {  
        "_parent": {  
            "type": "team"  
        },  
        "_routing": {  
            "required": true  
        },  
        "properties": {  
            "name": {  
                "type": "string"  
            }  
        }  
    }  
}
```

- + атомарность
- + дочерние документы возвращаются отдельно
- + быстрота поиска
- на одном шарде
- усложнение запроса (routing)

Ну что, посмотрим на наших чемпионов?



Установка и запуск ES

Конечно, Docker.

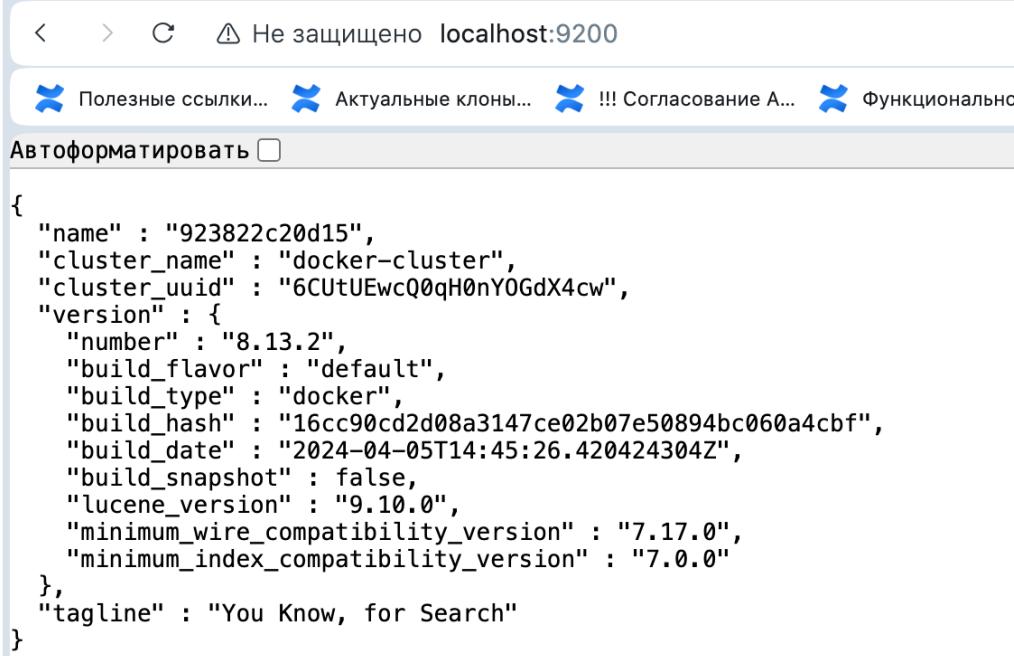
```
docker pull docker.elastic.co/elasticsearch/elasticsearch:8.13.2
```

...только не забудьте включить VPN 😊

Запускаем:

```
docker run --name es01 --net elastic -p 9200:9200 -it -m 1GB  
docker.elastic.co/elasticsearch/elasticsearch:8.13.2
```

Убеждаемся, что получилось...



⟨ ⟩ ⌂ ⚠ Не защищено localhost:9200

🔗 Полезные ссылки... 🔖 Актуальные клоны... 🔖 !!! Согласование А... 🔖 Функциональное

Автоформатировать

```
{  
  "name" : "923822c20d15",  
  "cluster_name" : "docker-cluster",  
  "cluster_uuid" : "6CUTUEwcQ0qH0nY0GdX4cw",  
  "version" : {  
    "number" : "8.13.2",  
    "build_flavor" : "default",  
    "build_type" : "docker",  
    "build_hash" : "16cc90cd2d08a3147ce02b07e50894bc060a4cbf",  
    "build_date" : "2024-04-05T14:45:26.420424304Z",  
    "build_snapshot" : false,  
    "lucene_version" : "9.10.0",  
    "minimum_wire_compatibility_version" : "7.17.0",  
    "minimum_index_compatibility_version" : "7.0.0"  
  "tagline" : "You Know, for Search"
```

Можно проверить, как он себя чувствует

GET https://localhost:9200/_cluster/health

Params Authorization Headers (7) Body Pre-request Script

Body Cookies Headers (3) Test Results

Pretty Raw Preview Visualize JSON ↻

```
1 "cluster_name": "docker-cluster",
2 "status": "yellow",
3 "timed_out": false,
4 "number_of_nodes": 1,
5 "number_of_data_nodes": 1,
6 "active_primary_shards": 6,
7 "active_shards": 6,
8 "relocating_shards": 0,
9 "initializing_shards": 0,
10 "unassigned_shards": 3,
11 "delayed_unassigned_shards": 0,
12 "number_of_pending_tasks": 0,
13 "number_of_in_flight_fetch": 0,
14 "task_max_waiting_in_queue_millis": 0,
15 "active_shards_percent_as_number": 66.66666666666666
16
17
```

GET https://localhost:9200/_cat/nodes

Params Authorization Headers (7) Body Pre-request Script Tests

Body Cookies Headers (3) Test Results

Pretty Raw Preview Visualize Text ↻

```
1 172.17.0.2 64 100 0 0.02 0.01 0.00 cdfhilmrstw * 923822c20d15
2
```

GET https://localhost:9200/_cat/shards

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Body Cookies Headers (3) Test Results

Pretty Raw Preview Visualize Text ↻

```
1 champions 0 p STARTED 6 21.9kb 21.9kb 172.17.0.2 923822c20d15
2 champions 0 r UNASSIGNED
3 .security-7 0 p STARTED 3 20.9kb 20.9kb 172.17.0.2 923822c20d15
4 .ds-logs-deprecation.elasticsearch-default-2024.04.15-000001 0 p STARTED 1 10.8kb 10.8kb 172.17.0.2 923822c20d15
5 champions_test 0 p STARTED 1 6.2kb 6.2kb 172.17.0.2 923822c20d15
6 champions_test 0 r UNASSIGNED
7 .ds-ilm-history-7-2024.04.15-000001 0 p STARTED 3 9.7kb 9.7kb 172.17.0.2 923822c20d15
8 champions_better 0 p STARTED 6 9.6kb 9.6kb 172.17.0.2 923822c20d15
9 champions_better 0 r UNASSIGNED
10
```

...или даже не проверять!

Потому что уже можно вносить документы!

The screenshot shows the Postman interface with a successful API call to `https://localhost:9200/champions_test/_doc/`. The request method is `POST`. The `Body` tab is selected, showing a JSON payload:

```
1 {  
2   "name": "Fernando Alonso",  
3   "team":  
4     {  
5       "name": "Renault",  
6       "engine": "Renault"  
7     }  
8 }
```

A red arrow points from the JSON response area to the right, indicating the successful creation of a document.

The response body is displayed in a pretty-printed JSON format:

```
1 {  
2   "_index": "champions_test",  
3   "_id": "fmKb5o4B1rkKUmzv2VJx",  
4   "_version": 1,  
5   "result": "created",  
6   "_shards": {  
7     "total": 2,  
8     "successful": 1,  
9     "failed": 0  
10   },  
11   "_seq_no": 0,  
12   "_primary_term": 1  
13 }
```

И... сразу их находить (например, по _id)!

GET https://localhost:9200/champions_test/_doc/fmKb5o4B1rkKUmzv2VJx

Params Authorization • Headers (7) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

Body Cookies Headers (3) Test Results

Pretty Raw Preview Visualize JSON 

```
1  "_index": "champions_test",
2  "_id": "fmKb5o4B1rkKUmzv2VJx",
3  "_version": 1,
4  "_seq_no": 0,
5  "_primary_term": 1,
6  "found": true,
7  "_source": {
8    "name": "Fernando Alonso",
9    "team": {
10      "name": "Renault",
11      "engine": "Renault"
12    }
13  }
14}
15}
```

Разными способами и видами

Весь индекс посмотреть

GET https://localhost:9200/champions/_search

Достать только _source:

GET https://localhost:9200/champions/_source/pCym4l4BxW9XFFzCsv7p

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Body Cookies Headers (3) Test Results

Pretty Raw Preview Visualize JSON

```
1 "name": "Max Verstappen",
2 "team": {
3     "engine": "RBPT Honda",
4     "name": "Red Bull"
5 },
6 "age": 26,
7 "famous_for": "Currently racing in dominant style. Mostly knows for the
8
9 
```

Добавить сразу пачку новых документов

POST https://localhost:9200/_bulk

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 { "index" : { "_index" : "champions" } }
2 { "name" : "Nico Rosberg", "age" : 39, "team" : { "name" : "Mercedes", "engine" : "Mercedes" } }
3 { "index" : { "_index" : "champions" } }
4 { "name" : "Jenson Button", "age" : 44, "team" : { "name" : "Brawn GP", "engine" : "Mercedes" } }
5 { "index" : { "_index" : "champions" } }
6 { "name" : "Sebastian Vettel", "age" : 36, "team" : { "name" : "Red Bull", "engine" : "Renault" } }
7 { "index" : { "_index" : "champions" } }
8 { "name" : "Max Verstappen", "age" : 26, "team" : { "name" : "Red Bull", "engine" : "RBPT Honda" } }
9 { "index" : { "_index" : "champions" } }
10 { "name" : "Lewis Hamilton", "age" : 39, "team" : { "name" : "Mercedes", "engine" : "Mercedes" } }
11
```

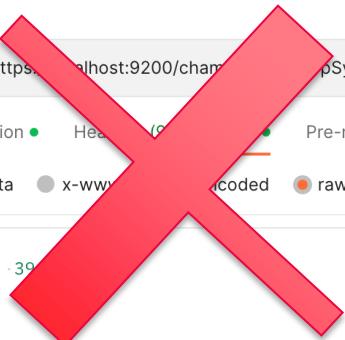
Или добавить одно поле в существующий

PUT https://localhost:9200/champions/_update/pSyv4l4BxW9XFFzC8P5I

Params Authorization Headers (9) Body Pre-request Script Tests

none form-data x-www-form-urlencoded raw binary GraphQL

```
1 {  
2   ... "age" : 39  
3 }
```



POST https://localhost:9200/champions/_update/pSyv4l4BxW9XFFzC8P5I

Params Authorization Headers (9) **Body** Pre-request Script Tests

none form-data x-www-form-urlencoded raw binary GraphQL

```
1 {  
2   ... "doc" :  
3     {  
4       ... "age" : 39  
5     }  
6 }
```

Или изменить значение поля по формуле

POST https://localhost:9200/champions/_update/pSyv4l4BxW9XFFzC8P5I

Params Authorization Headers (9) Body Pre-request Script Tests Se

none form-data x-www-form-urlencoded raw binary GraphQL

```
1 {  
2   "script" : {  
3     "source" : "ctx._source.age += params.count",  
4     "params" : {  
5       "count" : 1  
6     }  
7   }  
8 }
```

Или искать по конкретному полю документа

По точному значению

GET https://localhost:9200/champions/_search

Params Authorization Headers (9) Body Pre-request Script

none form-data x-www-form-urlencoded raw binary

```
1 {  
2   "query": {  
3     "match": {  
4       "name": "Niko"  
5     }  
6   }  
7 }
```

С учётом опечаток

GET https://localhost:9200/champions/_search

Params Authorization Headers (9) Body Pre-rec

none form-data x-www-form-urlencoded raw

```
1 {  
2   "query": {  
3     "match": {  
4       "name": {  
5         "query": "Nico",  
6         "fuzziness": "AUTO"  
7       }  
8     }  
9   }  
10 }
```

Или другими разными способами

По поддокументу

GET https://localhost:9200/champions/_search

Params Authorization Headers (9) Body Pre-request

none form-data x-www-form-urlencoded raw

```

1 {
2   "query": {
3     "match": {
4       "team.name": "Mercedes"
5     }
6   }
7 }
```

С учётом диапазона

GET https://localhost:9200/champions/_search

Params Authorization Headers (9) Body Pre-request

none form-data x-www-form-urlencoded raw

```

1 {
2   "query": {
3     "range": {
4       "age": {
5         "gte": 20,
6         "lt": 40
7       }
8     }
9   }
10 }
```

Или по маске

GET https://localhost:9200/champions/_search

Params Authorization Headers (9) Body Pre-request

none form-data x-www-form-urlencoded raw

```

1 {
2   "query": {
3     "wildcard": {
4       "name": {
5         "value": "*r*"
6       }
7     }
8   }
9 }
```

Или добавить новое поле в документы

POST https://localhost:9200/_bulk

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 { "update" : { "_id" : "pSv4I4BxW9XFFzC8P5I", "_index" : "champions" } }
2 { "doc" : { "famous_for" : "Have the same amount of titles like Michael Schumacher - seven. Now have a little more." }
3 { "update" : { "_id" : "pCym4I4BxW9XFFzCsv7p", "_index" : "champions" } }
4 { "doc" : { "famous_for" : "Currently racing in dominant style. Mostly knows for the youngest debut in Formula 1." }
5 { "update" : { "_id" : "qyyu4Y4BxW9XFFzC3f7Z", "_index" : "champions" } }
6 { "doc" : { "famous_for" : "Mostly know for his fight for title with Lewis Hamilton for consequently the Formula 1 world championship." }
7 { "update" : { "_id" : "rCyu4Y4BxW9XFFzC3f7Z", "_index" : "champions" } }
8 { "doc" : { "famous_for" : "Made his title in the almost unique circumstances when his team gone to bankruptcy." }
9 { "update" : { "_id" : "rSyu4Y4BxW9XFFzC3f7Z", "_index" : "champions" } }
10 { "doc" : { "famous_for" : "Right now retired and taking his time for the ecology themes. Most like bees and butterflies." }
11 { "update" : { "_id" : "piyw4I4BxW9XFFzCUf5r", "_index" : "champions" } }
12 { "doc" : { "famous_for" : "Mostly known for his commitment for races. He became the champion for the first time in 2004." }
13 { "update" : { "_id" : "pCym4I4BxW9XFFzCsv7p", "_index" : "champions" } }
```

Или удалить поле из документов

Поштучно

POST https://localhost:9200/champions/_update/qyyu4Y4BxW9XFFzC3f7Z

Params Authorization • Headers (9) Body • Pre-request Script Tests Set

none form-data x-www-form-urlencoded raw binary GraphQL

```
1 {  
2   "script" : "ctx._source.remove('famous_for')"  
3 }  
4 }
```

Или из всех сразу

POST https://localhost:9200/champions/_update_by_query

Params Authorization • Headers (9) Body • Pre-request Script

none form-data x-www-form-urlencoded raw binary

```
1 {  
2   "script" : {  
3     "source" : "ctx._source.remove('famous_for')"  
4   },  
5   "query" : {  
6     "match_all" : {}  
7   }  
8 }  
9 }
```

Если загрузили по SQL, то – он тоже есть!

The screenshot shows a Postman API request for `https://localhost:9200/_sql?format=txt`. The `Body` tab contains the following SQL query:

```
1
2 "query": "SELECT name, age, team.name as \"team name\", team.engine as \"team engine\", substring(famous_for, 0, 90) as short_famous_for FROM champions"
3
```

The response body displays the results of the query in a tabular format:

	name	age	team name	team engine	short_famous_for
3	Max Verstappen	26	Red Bull	RBPT Honda	Currently racing in dominant style. Mostly knows for the youngest debut in Formula 1. Afte
4	Niko Rosberg	39	Mercedes	Mercedes	Mostly know for his fight for title with Lewis Hamilton for consequently the three seasons
5	Jenson Button	44	Brawn GP	Mercedes	Made his title in the almost unique circumstances when his team gone to bankruptcy and was
6	Sebastian Vettel	36	Red Bull	Renault	Right now retired and taking his time for the ecology themes. Most like bees and garbage c
7	Fernando Alonso	43	Aston Martin	Mercedes	Mostly known for his commitment for races. He became the champion for the first time in 20
8	Lewis Hamilton	39	Mercedes	Mercedes	Have the same amount of titles like Michael Schumacher - seven. Now have a less success pe

В чём сила этой магии?

ES всё сделал за нас, по одному документу. Маппинг, настройки, вообще всё. You know, for Search

```
GET https://localhost:9200/champions_test/
Params Authorization Headers (7) Body Pre-request Script Tests Settings
Headers ⚡ Hide auto-generated headers
Body Cookies Headers (3) Test Results
Pretty Raw Preview Visualize JSON ▾
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
{
  "champions_test": {
    "aliases": {},
    "mappings": {
      "properties": {
        "name": {
          "type": "text",
          "fields": {
            "keyword": {
              "type": "keyword",
              "ignore_above": 256
            }
          }
        },
        "team": {
          "properties": {
            "engine": {
              "type": "text",
              "fields": {
                "keyword": {
                  "type": "keyword",
                  "ignore_above": 256
                }
              }
            },
            "name": {
              "type": "text",
              "fields": {
                "keyword": {
                  "type": "keyword",
                  "ignore_above": 256
                }
              }
            }
          }
        }
      },
      "settings": {
        "index": {
          "routing": {
            "allocation": {
              "include": {
                "_tier_preference": "data_content"
              }
            }
          },
          "number_of_shards": "1",
          "provided_name": "champions_test",
          "creation_date": "1713265957131",
          "number_of_replicas": "1",
          "uuid": "httpsDIPSPULW2nK47hNww",
          "version": {
            "created": "8503860"
          }
        }
      }
    }
  }
}
```

А если нам что-то не подошло?

Например, мы хотим добавить какой-то дополнительный анализатор на наши поля (предположим, для русского текста)

Или разметить через mappings поля в наших документах несколько иначе, чем за нас решил ES?

Тогда нас ждут боль и страдания

You cannot change the mapping (including the analyzer) of an existing field. What you need to do if you want to change the mapping of existing documents is reindex those documents to another index with the updated mapping

(цитата из официальной документации)



Как это преодолеть?

Подготовить новый индекс, настроить маппинг под наши изменившиеся задачи, после чего сделать реиндекс:

```
POST https://localhost:9200/_reindex

Params: Authorization • Headers (9) • Body •
none form-data x-www-form-urlencoded

{
  "source": {
    "index": "champions"
  },
  "dest": {
    "index": "champions_better"
  }
}
```

...но есть и плюс – все документы, перемещённые в новый индекс, сохраняют все свои атрибуты

Сложности с транзакциями в ES



Преимущества ES

- Скорость работы
- Масштабируемость
- Отказоустойчивость
- Гибкий и мощный язык запросов
- Полнотекстовый поиск и анализ
- Своя экосистема (ES + Logstash + Kibana)

Недостатки ES

- Сложность масштабирования
- Высокие требования к ресурсам
- Сложность запросов и поиска
- Отсутствие встроенной поддержки для транзакций
- Сложность обновления и удаления данных
- ~~Отсутствие встроенной безопасности~~ (с 8-й версии есть!)

Где пригодится ES?

- Поиск (полнотекстовый)
- Анализ логов
- Хранение и визуализация мониторинга и метрик
- Рекомендательные системы
- Интеллектуальный поиск
- eCommerce

В качестве примера – eCommerce на PG

- + JSON в качестве атрибута (с CONSTRAINT на поле внутри)
- + есть текстовый поиск «из коробки» (ts_query)
- + есть хранимые процедуры
- Это всё равно SQL (надо думать над схемой, связями)
- не все ORM нормально понимают JSON
- работать будет не очень быстро

В качестве примера – eCommerce на Mongo

- + схемы нет (можно не думать?)
- + лёгкая масштабируемость
- + никаких вам JOIN-ов
- валидация типов в коде
- нет хранимых процедур
- не очень удобный синтаксис запросов

В качестве примера – eCommerce на ES

- + всё через REST
- + валидация на уровне хранилища (mappings)
- + не нужен ORM (см. п.1)
- + JSON-query ещё более декларативен, чем SQL
- нет транзакций
- требователен к железу
- запросы декларативны, но при этом сложны
- сложности с обновлением данных (зависит от структуры)

Спасибо за внимание!

www.ifmo.ru

