# Tools для веб архитектур

## Elastic search

сайт

[Elasticsearch: The Official Distributed Search & Analytics Engine | Elastic](https://www.elastic.co/elasticsearch)

облачные услуги (моя личная среда)

<https://37786ab484b645808192ed124d462b99.us-central1.gcp.cloud.es.io/app/elasticsearch/overview>

### Инфо

Elasticsearch — это документо-ориентированная поисковая система с API-интерфейсом, работающая поверх движка Apache Lucene.

В отличие от реляционных баз (типа MySQL или PostgreSQL), здесь всё строится вокруг документов в формате JSON, а не таблиц.

#### Как это работает

##### Добавление документов

1. Получение и распределение запроса

1.1. REST API и Coordinating Node

* **REST API:** Клиент отправляет JSON-документ через HTTP-запрос (например, с помощью curl или другого инструмента).
* **Coordinating Node:** Узел, принимающий запрос, выполняет роль маршрутизатора. Он:  
  + Принимает запрос.
  + Выясняет, на какой шард (часть индекса) нужно сохранить документ согласно стратегии распределения.
  + Проводит предварительную валидацию и, при необходимости, применяет фильтры безопасности или трансформации запроса.

*Ссылки:*

* [Elasticsearch REST API](https://www.elastic.co/guide/en/elasticsearch/reference/current/rest-apis.html)

2. Определение места хранения: индекс, шард и реплика

2.1. Индекс и маппинг

* **Индекс:** Логическая структура, похожая на таблицу в реляционной базе данных. Каждому индексу можно задать маппинг, который определяет типы и настройки полей (например, text для полнотекстового анализа, keyword для точных значений, date, float и т.д.).
* **Маппинг:** Определяет, как будут анализироваться и храниться поля документа. Например, поле name типа text будет обрабатываться анализатором для разбиения на токены.

*Ссылки:*

* [Mapping в Elasticsearch](https://www.elastic.co/guide/en/elasticsearch/reference/current/mapping.html)

2.2. Шард и репликация

* **Шард:** Каждый индекс разбивается на один или несколько шард, чтобы распределять нагрузку и обеспечить параллельную обработку запросов.
* **Реплика:** Копии шард, которые используются для отказоустойчивости и повышения производительности чтения.

3. Обработка документа: Анализ и индексирование

3.1. Анализ документа

* **Анализатор (Analyzer):** Для полей типа text Elasticsearch применяет анализатор, который состоит из:  
  + **Char Filters:** Предварительная обработка текста (удаление HTML-тегов, нормализация символов).
  + **Tokenizer:** Разбивает строку на токены (например, слово «Elasticsearch Tutorial» превращается в токены ["elasticsearch", "tutorial"]).
  + **Token Filters:** Применяются фильтры для нормализации (приведение к нижнему регистру, удаление стоп-слов, стемминг).

В результате исходный JSON-документ преобразуется:  
 Исходное поле:

json

КопироватьРедактировать

"name": "Elasticsearch Tutorial"

Преобразуется в список токенов:

json

КопироватьРедактировать

"tokens": ["elasticsearch", "tutorial"]

*Ссылки:*

* [Анализаторы в Elasticsearch](https://www.elastic.co/guide/en/elasticsearch/reference/current/analysis.html)

3.2. Построение инвертированного индекса (Inverted Index)

* **Inverted Index:** После анализа для каждого токена создается структура, связывающая термин с идентификаторами документов (и позициями токенов внутри документа).

Пример (псевдо-JSON представление):  
  
 json  
КопироватьРедактировать  
{

"inverted\_index": {

"elasticsearch": [{ "doc\_id": 1, "positions": [1] }],

"tutorial": [{ "doc\_id": 1, "positions": [2] }]

}

}

* Это позволяет при поиске быстро определить, в каких документах содержится запрошенный термин.

4. Физическое хранение: Lucene и сегменты

4.1. Роль Apache Lucene

Apache Lucene: Ядро, отвечающее за построение и хранение полнотекстовых индексов. Elasticsearch использует Lucene для:

Хранения инвертированного индекса.

Организации данных в сегменты.

Выполнения высокопроизводительного поиска и агрегаций.

*Ссылки:*

[Apache Lucene](https://lucene.apache.org/core/)

4.2. Сегменты и их структура

Сегменты: Каждый шард в Elasticsearch представлен набором сегментов Lucene. Сегменты – это неизменяемые структуры, содержащие:

Инвертированный индекс: Хранит токены и списки документов (postings lists).

Stored Fields: Хранят исходные значения полей, необходимые для возврата данных клиенту.

Doc Values: Для сортировки и агрегаций.

Процесс записи:

Документ сначала поступает в оперативную память и транзакционный журнал (transaction log).

На основе буферов формируются маленькие сегменты.

По мере накопления происходит слияние сегментов, когда несколько маленьких сегментов объединяются в один более крупный для оптимизации поиска.

*Ссылки:*

[Lucene Index File Formats](https://lucene.apache.org/core/8_11_0/core/org/apache/lucene/codecs/lucene50/package-summary.html) – информация о форматах файлов

[Segment Merging](https://www.elastic.co/guide/en/elasticsearch/reference/current/index-modules.html#_index_merging)

4.3. Запись на диск

Сегменты сохраняются как бинарные файлы на жестком диске.

Memory-Mapped Files (mmap): Часто используются для ускорения доступа к сегментам, позволяя операционной системе эффективно кэшировать данные.

5. Итоговый процесс от начала до конца

**Получение запроса:** Клиент отправляет JSON-документ через REST API.  
 Пример запроса:  
  
 bash  
КопироватьРедактировать  
curl -X POST "localhost:9200/products/\_doc/1" -H 'Content-Type: application/json' -d '{

"name": "Elasticsearch Tutorial",

"price": 999,

"category": "smartphones",

"created\_at": "2024-04-15"

}'

1. **Распределение запроса:** Coordinating Node определяет, к какому шард следует отправить документ согласно стратегиям распределения и маппингу индекса.
2. **Анализ и обработка:**
   * Поля типа text обрабатываются анализаторами.
   * Исходный текст превращается в набор токенов.
   * Формируется инвертированный индекс, где для каждого токена создаётся запись с идентификаторами и позициями документа.
3. **Формирование сегментов:**
   * Обработанные данные попадают в оперативную память и транзакционный лог.
   * Из буферов формируются небольшие сегменты Lucene.
4. **Сохранение данных на диск:**
   * Сегменты записываются как бинарные файлы на Data Node (дисковое хранилище).
   * Происходит слияние сегментов для повышения эффективности поиска.
5. **Дальнейшая работа:**
   * Данные доступны для поиска с помощью инвертированного индекса.
   * Запросы на полнотекстовый поиск обращаются к postings list, что позволяет быстро находить документы.

Полезные источники для углубления знаний

* [Elasticsearch: The Definitive Guide](https://www.elastic.co/guide/en/elasticsearch/guide/current/index.html)
* [Understanding Lucene's Inverted Index](https://www.elastic.co/blog/understanding-lucene-inverted-index)
* [Elasticsearch Analysis and Analyzers](https://www.elastic.co/guide/en/elasticsearch/reference/current/analysis.html)

#### Индекс Lucene

Так называется структура данных которая создаётся при добавлении документов в elastic search Lucene

Сегмент

**Внутри сегментов Lucene содержатся:**

* **Инвертированный индекс:** Для каждого текстового поля (например, «name») анализатор разбивает исходный текст на токены (слова).  
   Для каждого токена создаётся запись (postings list), которая указывает, в каких документах и на каких позициях он встречается.  
   Это позволяет проводить быстрый полнотекстовый поиск.
* **Doc Values:** Это колонно-ориентированные структуры, предназначенные для быстрого доступа при сортировке и агрегациях.  
   Они сохраняют значения полей в оптимизированном для чтения формате.
* **Stored Fields:** Некоторые поля могут сохраняться отдельно (как есть) для возврата данных клиенту при поиске, без необходимости воссоздавать их из инвертированного индекса.

#### Инвертированные индексы

#### Сравнение с базой данных

Однако, Elasticsearch не является реляционной БД. Он оптимизирован для поиска и анализа данных с фокусом на скорость обработки больших объёмов информации, а не на строгую нормализацию и поддержание связей между таблицами.

### Терминал

На последних версиях нужно:

1) делать запрос только через защищенное соединение https

2) И с аргументом -k (позволяет согласится на небезопасный запрос)

Самые основные команды

<https://www.elastic.co/guide/en/elasticsearch/guide/current/_talking_to_elasticsearch.html>

curl -X GET "https://localhost:9200/" -u elastic:password -k - Базовый запрос

#### Работа с эластик сёрчем

Создание индекса

curl -X PUT "<https://localhost:9200/products>" -u elastic:password -k

Создание индекса с маппингом

curl -X PUT "[https://](https://localhost:9200/products)localhost:9200/products" -u elastic:password -k -H 'Content-Type: application/json' -d '

{

"mappings": {

"properties": {

"name": {

"type": "text"

},

"price": {

"type": "float"

},

"category": {

"type": "keyword"

},

"created\_at": {

"type": "date"

}

}

}

}'

Добавление маппинга к существующему индексу

curl -X PUT "https://localhost:9200/products/\_mapping" -u elastic:password -k -H 'Content-Type: application/json' -d '

{

"properties": {

"name": {

"type": "text"

},

"price": {

"type": "float"

},

"category": {

"type": "keyword"

},

"created\_at": {

"type": "date"

}

}

}'

Добавление документа в индекс

curl -X POST "https://localhost:9200/products/\_doc/1" -u elastic:password -k -H 'Content-Type: application/json' -d '

{

"name": "iPhone 14",

"price": 999,

"category": "smartphones",

"created\_at": "2024-04-15"

}'

Поиск по индексу документа

curl -X GET "<https://localhost:9200/products/_doc/1>" -u elastic:password -k

Поиск по значению поля документа

curl -X GET "https://localhost:9200/products/\_search" \

-u elastic:password \

-k \

-H 'Content-Type: application/json' \

-d '

{

"query": {

"match": {

"name": "iPhone 14"

}

}

}'

Поиск по значению поля:

curl -X GET "https://localhost:9200/products/\_search" \

-u elastic:password \

-k \

-H 'Content-Type: application/json' \

-d '

{

"query": {

"match": {

"name": "iPhone 14"

}

}

}'

Фильтр:

curl -X GET "[https://](https://localhost:9200/products/_doc/1)localhost:9200/products/\_search" -u elastic:password -k -H 'Content-Type: application/json' -d '

{

"query": {

"range": {

"price": {

"gte": 500,

"lte": 1000

}

}

}

}'

Запрос по значению поля + фильт:

curl -X GET "[https://](https://localhost:9200/products/_doc/1)localhost:9200/products/\_search" \

-u elastic:password \

-k \

-H 'Content-Type: application/json' \

-d '

{

"query": {

"bool": {

"must": [

{

"match": {

"name": "iPhone 14"

}

},

{

"range": {

"price": {

"gte": 500,

"lte": 1000

}

}

}

]

}

}

}'

удаление документа

curl -X DELETE "[https://](https://localhost:9200/products/_doc/1)localhost:9200/products/\_doc/1" -u elastic:password -k

удаление индекса

curl -X DELETE "[https:/](https://localhost:9200/products/_doc/1)/localhost:9200/products" -u elastic:password -k

#### Поиск по тексту

Полнотекстовый поиск (можно искать по отдельным словам):

curl -X GET "https://localhost:9200/products/\_search" \

-u elastic:password \

-k \

-H 'Content-Type: application/json' \

-d '

{

"query": {

"match": {

"name": "iPhone"

}'} }

Нечёткий поиск (с ошибками):

curl -X GET "https://localhost:9200/products/\_search" \

-u elastic:password \

-k \

-H 'Content-Type: application/json' \

-d '

{

"query": {

"match": {

"name": {

"query": "iPhane",

"fuzziness": "AUTO"

}

}

}

}'

## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Архитектуры с Elastic

Для обеспечения надежности транзакций обычно используют комбинированные архитектуры с Elastic search и реляционной БД

Примеры архитектур

1. **Система-источник (Source of Truth) плюс поисковый движок** В этой схеме база данных является основным хранилищем данных. Все операции записи, обновления и удаления происходят в БД, а затем данные реплицируются или индексируются в Elasticsearch.  
   * **Репликация данных:** Используется механизм синхронизации или обмен сообщений (например, через очереди, Change Data Capture), чтобы изменения в БД сразу отражались в индексе Elasticsearch.
   * **Отложенная синхронизация:** При некоторых сценариях допускается задержка между обновлением в БД и обновлением в Elasticsearch, если скорость поиска не критична.
2. **Дублирование функций для повышения масштабируемости** Некоторые системы используют Elasticsearch даже для операций аналитики или отчетности, оставляя транзакционные операции в БД. Это позволяет разгрузить основное хранилище данных и улучшить масштабируемость системы.

**Кеширование сложных поисковых запросов** Elasticsearch может выступать как высокопроизводительный слой для поиска, избавляя базу данных от избыточной нагрузки на выполнение сложных запросов, особенно в сценариях, когда требуется быстрый отклик для пользователей.