# **Terminologies**

### **Core Terminology – Elasticsearch vs RDBMS**

To bridge the gap between traditional relational databases (like MySQL) and Elasticsearch, here’s a quick comparison of common terms:

| ****Elasticsearch**** | ****RDBMS (e.g., MySQL)**** | ****MongoDB**** |
| --- | --- | --- |
| Index | Table | Collection |
| Document | Row / Record | Document |
| Field | Column | Field |

* An **index** is a collection of related documents (think: a table in RDBMS).
* A **document** is a single record (like a row).
* A **field** is a data attribute within a document (similar to a column).

**Note:**  
The term **"index"** is also used as a **verb** in Elasticsearch. For example, “index the documents” means to store and organize them for efficient searching.

Some people use "indices" (the correct plural) and others say "indexes"—both are commonly accepted.

**Open: http://localhost:5601/app/dev\_tools**

### 1. **Index – Create / Delete**

# Create an index named "blog"

PUT /blog

# Delete the "blog" index

DELETE /blog

### 2. **CRUD – Adding Documents (Auto ID)**

# Add a document without specifying an ID (auto-generated)

POST /blog/\_doc

{

"title": "Getting started with Elasticsearch",

"author": "Samiun",

"tags": ["search", "elasticsearch"],

"published": "2025-07-04"

}

### 3. **Source Meta Data Fields**

# View internal metadata (\_id, \_index, etc.)

GET /blog/\_search

{

"\_source": false

}

### 4. **CRUD – Adding Documents with ID**

# Add a document with a custom ID

PUT /blog/\_doc/1

{

"title": "Mastering Elasticsearch",

"author": "Samiun",

"tags": ["advanced", "lucene"],

"published": "2025-07-01"

}

### 5. **CRUD – Querying Documents**

# Match all query

GET /blog/\_search

{

"query": {

"match\_all": {}

}

}

# Match query

GET /blog/\_search

{

"query": {

"match": {

"title": "elasticsearch"

}

}

}

### 6. **CRUD – Updating Documents**

# Full document replacement (PUT)

PUT /blog/\_doc/1

{

"title": "Updated Title",

"author": "Samiun",

"tags": ["update"],

"published": "2025-07-04"

}

### 7. **CRUD – Patch (Partial Update)**

# Update part of a document (PATCH-style with POST)

POST /blog/\_update/1

{

"doc": {

"tags": ["update", "patched"]

}

}

### 8. **CRUD – Scripted Update**

# Use a script to update field value

POST /blog/\_update/1

{

"script": {

"source": "ctx.\_source.views = params.count",

"lang": "painless",

"params": {

"count": 150

}

}

}

### 9. **CRUD – Delete**

# Delete a document

DELETE /blog/\_doc/1

### 10. **Inverted Index (Conceptual)**

Not a direct API, but you can view analysis results to see how text is tokenized (stored in the inverted index).

# Analyze a text field

POST /\_analyze

{

"analyzer": "standard",

"text": "Elasticsearch is fast and scalable"

}

### 11. **Term Dictionary / Term Frequency / Document Frequency**

# Term vector API: show term frequency, positions, offsets, etc.

POST /blog/\_termvectors/1

{

"fields": ["title"],

"term\_statistics": true,

"field\_statistics": true

}

### 12. **OPTIONAL – Segments**

# Show segment information for an index

GET /blog/\_segments

### 13. **OPTIONAL – Refresh API**

# Force refresh to make documents searchable immediately

POST /blog/\_refresh

### Tip:

If you want a fresh index for experimentation:

DELETE /blog

PUT /blog

## Summary: Elasticsearch CRUD & Internals via Kibana Dev Tools

This practical session covers core Elasticsearch operations using **Kibana Dev Tools**, including creating indexes, inserting documents, querying data, and understanding the inner workings of Elasticsearch like term dictionaries and segments.

### Key Takeaways:

| Topic | What You Learned |
| --- | --- |
| **Index Operations** | How to create and delete an index |
| **CRUD – Add** | Adding documents with and without IDs |
| **Metadata & Querying** | Accessing source fields and searching documents with match, match\_all |
| **CRUD – Update** | Updating documents completely, partially (patch), or via scripting |
| **CRUD – Delete** | How to remove a document |
| **Inverted Index** | Text analysis with the \_analyze API showing how Elasticsearch tokenizes text |
| **Term Stats** | Use of \_termvectors API to inspect term frequency, document frequency |
| **Segments** | Understanding segment structure using \_segments API |
| **Refresh API** | Manually refreshing an index to make documents searchable immediately |

Real-world use case:

You can imagine building a blog app where each blog post is indexed and searchable by tags, author, or title. You update view counts using scripted updates and monitor term frequencies for SEO optimization.

## QUIZ – Test Your Knowledge

Choose the correct answers.

**1. Which method is used to create an index?**

a. POST /index\_name  
b. PUT /index\_name  
c. GET /index\_name  
d. PATCH /index\_name

**Correct Answer:** **b.** PUT /index\_name

**2. Which API returns a breakdown of how Elasticsearch tokenizes and analyzes text?**

a. \_doc  
b. \_search  
c. \_analyze  
d. \_mapping

**Correct Answer:** **c.** \_analyze

**3. What will the** \_source: false **setting do in a search query?**

a. Return only \_id  
b. Return all metadata fields only  
c. Hide original document content  
d. Disable indexing

**Correct Answer:** **c.** Hide original document content

**4. What is the purpose of the** \_termvectors **API?**

a. Create a vector index  
b. Fetch term frequency and statistics  
c. Index vector documents  
d. Convert text into synonyms

**Correct Answer:** **b.** Fetch term frequency and statistics

**5. Which field is updated in this script?**

"script": {

"source": "ctx.\_source.views += 1"

}

a. \_id  
b. views  
c. title  
d. tags

**Correct Answer:** **b.** views

**6. What does the** \_refresh **API do?**

a. Deletes old documents  
b. Clears cache  
c. Makes recently indexed documents searchable immediately  
d. Optimizes query speed

**Correct Answer:** **c.** Makes recently indexed documents searchable immediately