# 

# Understanding Inverted Index in Elasticsearch

## 1. Introduction

Jatin: As now we now the basics of Text Analysis, let’s look at what happens with the results, being tokens.   
In this lecture, we delve into the concept of an inverted index, one of the key data structures used in Elasticsearch to ensure efficient data retrieval. This understanding forms the foundation for performing fast and effective search operations.

## 2. Data Structures in Elasticsearch

1. Elasticsearch uses **different data structures** depending on the field's data type.

2. The purpose of using multiple data structures is to optimize data retrieval for various **access patterns**:  
 - Searching for terms is different from aggregating data.

3. The primary **data structures are managed by Apache Lucene**, not Elasticsearch.

## 3. Introduction to Inverted Index

1. An inverted index is a mapping of terms (tokens) to the documents containing those terms.  
2. Tokens are generated during the analysis process and are stored in an inverted index as terms.

3. Instead of mapping documents to their terms, an inverted index inverts this relationship for efficient lookups.

## 4. Example of an Inverted Index

1. Consider the text 'Why did the third duck cross the road?' indexed into a single document.  
 - Each unique term is stored in the index with references to the document IDs containing those terms.

2. Adding more documents expands the inverted index with more terms and document references.

3. **Example**: Searching for the term 'ducks' returns documents #1 and #3.

## 5. Efficiency of Inverted Index

1. The inverted index allows efficient lookups by mapping terms to document IDs.  
2. This design is faster than a forward mapping (documents to terms) for search queries.

3. Terms are sorted alphabetically within the inverted index, enabling optimized searches.

## 6. Additional Information Stored in Inverted Index

1. Beyond term-to-document mapping, the inverted index includes data for relevance scoring:  
 - This helps rank documents by how well they match a search query.

2. Relevance scoring will be explored in detail later in the course.

## 7. Multiple Fields and Inverted Indices



1. Each text field in a document has its own inverted index.  
2. Example: A document with 'name' and 'description' fields will have two inverted indices, one for each field.

3. Text is analyzed and indexed for each field independently, ensuring efficient searches across fields as different data

## 8. Other Data Structures in Elasticsearch

1. While text fields use inverted indices, other field types use different data structures:  
 - Numeric and geospatial data: Stored as BKD trees, optimized for range and geospatial queries.  
 - Date fields: Internally stored as long values and also use BKD trees.

2. These data structures will be covered in more detail later in the course.

## 9. Summary

1. An inverted index maps terms to document IDs, enabling fast term lookups and efficient search queries.

2. Each text field has its own inverted index, while other field types use data structures like BKD trees.

3. The inverted index is managed by Apache Lucene, which Elasticsearch builds upon.

4. This foundational concept of inverted indices ensures high performance in Elasticsearch's search capabilities.  
  
