ELASTICSEARCH

Presentation by AmirHossein Rahmati

Modern Information Retrieval Course lectured by Dr. Soleymani | 2024 | Sharif University of Technology

TOPICS COVERED

Basic Knowledge of Elasticsearch and Its Applications:

- Introduction to Elasticsearch and its use cases.
- Overview of how Elasticsearch fits into various applications such as real-time search, log analysis, and more.

Working with Elasticsearch on the Data of Phase One of the Project and Familiarizing with the Types of Queries:

- Hands-on demonstration of working with Elasticsearch on sample data.

Basic Familiarity with Distributed Concepts such as Cluster, Node, Shard, Replica in Elasticsearch:

- Understanding the distributed nature of Elasticsearch.
- Explanation of cluster, node, shard, and replica concepts and their significance in Elasticsearch architecture.

Overview of Document Indexing Concepts in Information Retrieval:

- Explanation of document indexing process in Elasticsearch.
- Importance of indexing for efficient search and retrieval.

Getting to Know Apache Lucene as an Elasticsearch Search Engine and In-depth exploration of document indexing process in Apache Lucene:

- Discussion on indexing strategies and best practices.

Getting to Know Scoring in Elasticsearch (Apache Lucene):

- Explanation of scoring mechanism in Elasticsearch.

ELASTICSEARCH AND ITS APPLICATIONS

Introduction to Elasticsearch:

- Elasticsearch is a distributed search and analytics engine built on top of Apache Lucene.
- It is designed for scalability, reliability, and real-time search capabilities.

Use Cases of Elasticsearch:

- Real-Time Search.
- Log Analysis.
- Full-Text Search.

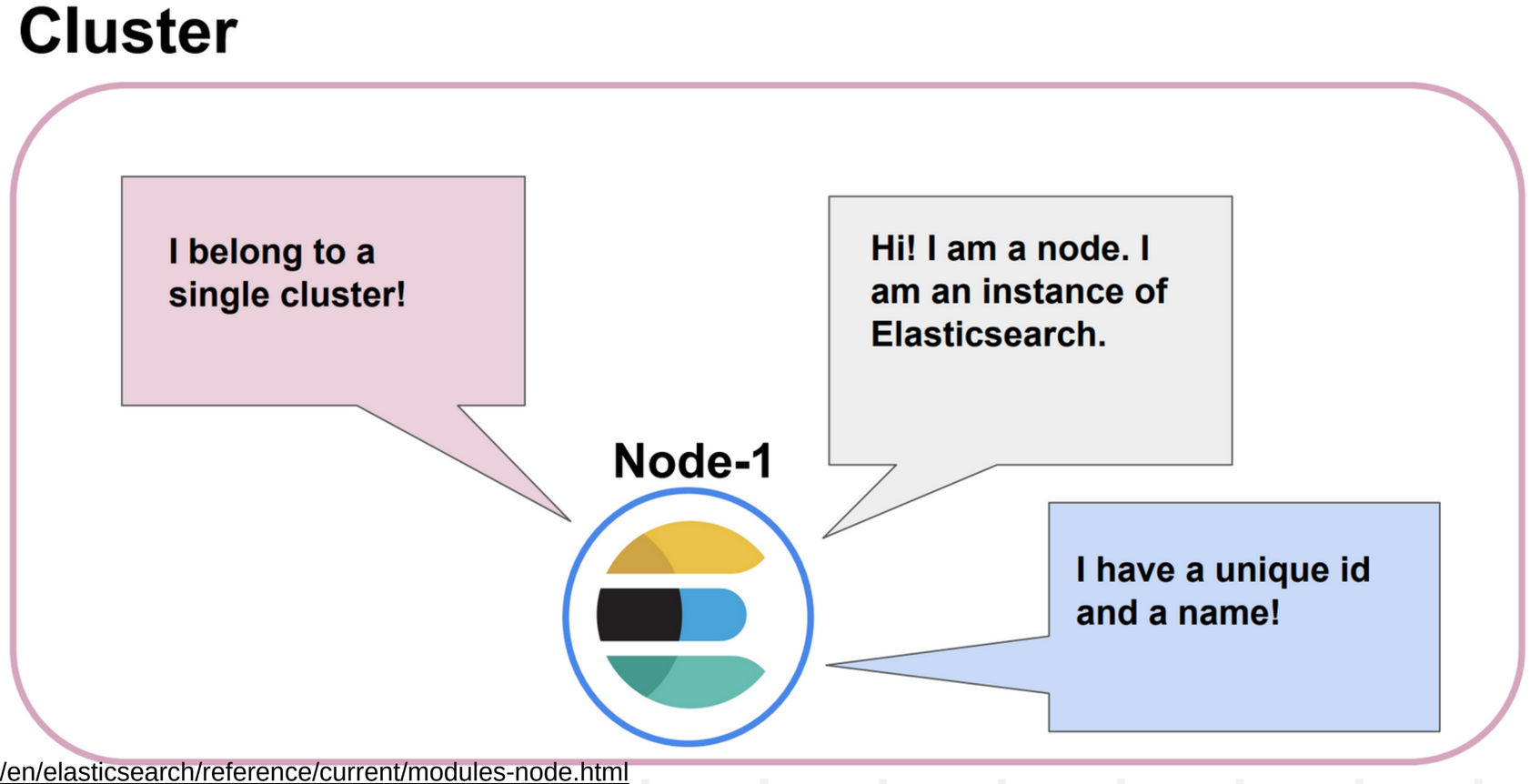
https://www.elastic.co/blog/found-uses-of-elasticsearch

WORKING WITH ELASTICSEARCH

Elastic Cloud:
https://www.elastic.co/cloud

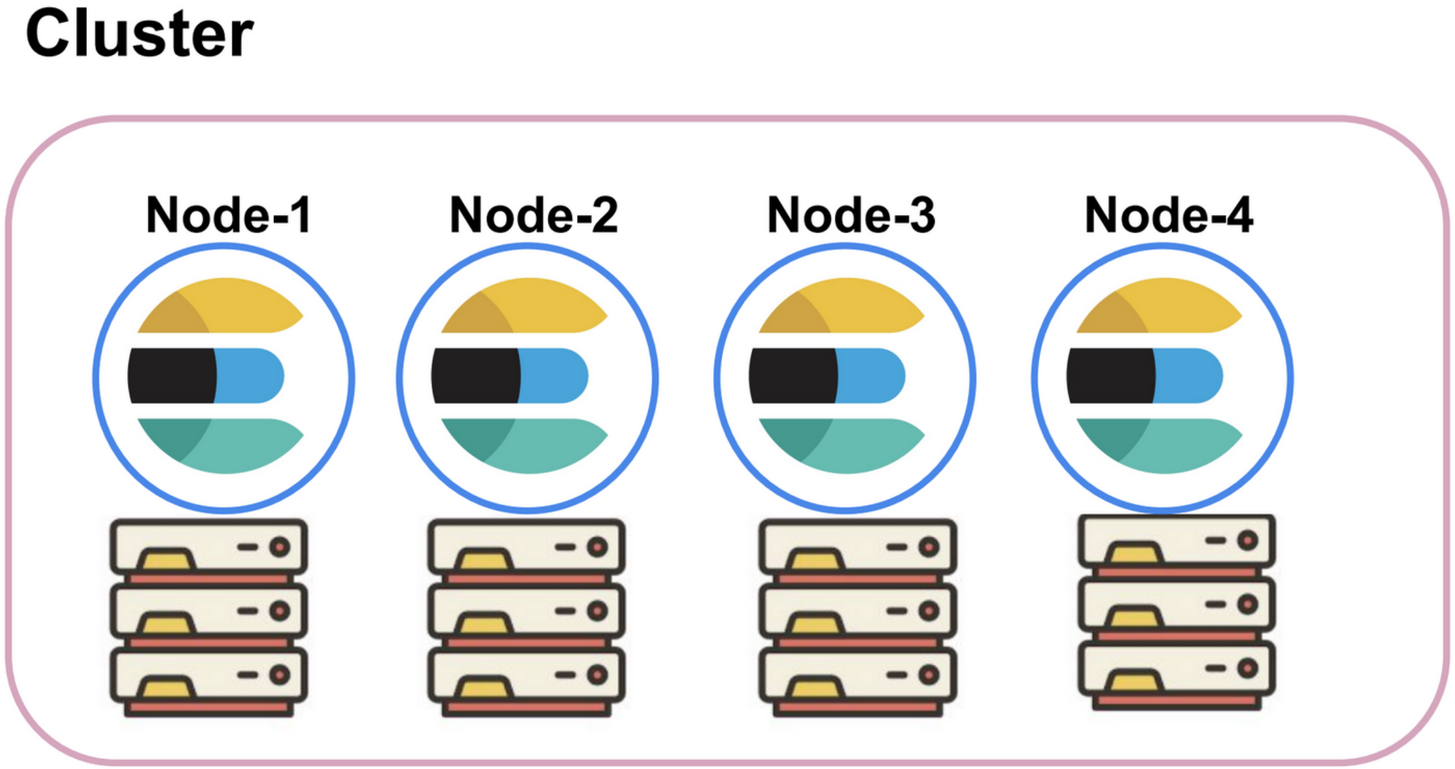
API links:
https://www.elastic.co/guide/en/elasticsearch/reference/current/indices-create-index.html
https://www.elastic.co/guide/en/elasticsearch/reference/current/docs-index_.html
https://www.elastic.co/guide/en/elasticsearch/reference/current/docs-bulk.html
https://www.elastic.co/guide/en/cloud/current/ec-manage-kibana-settings.html
https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-mlt-query.html
https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-term-query.html
https://www.elastic.co/guide/en/elasticsearch/reference/current/docs-termvectors.html
https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-bool-query.html
https://www.elastic.co/guide/en/elasticsearch/reference/current/query-filter-context.html
https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-range-query.html
https://github.com/LisaHJung/Beginners-Crash-Course-to-Elastic-Stack-Series-Table-of-Contents?tab=readme-ov-file

DISTRIBUTED CONCEPTS IN ELASTICSEARCH



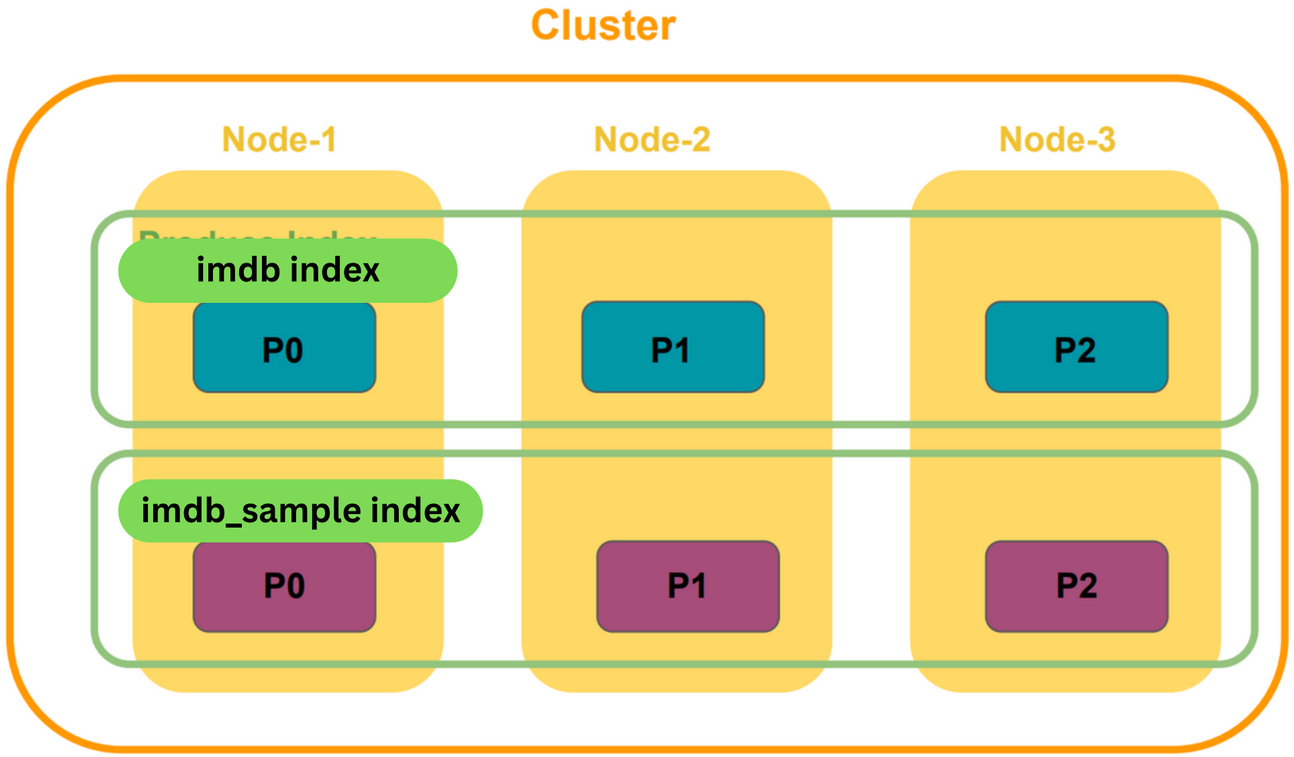
https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-node.html

DISTRIBUTED CONCEPTS IN ELASTICSEARCH



https://opster.com/guides/elasticsearch/high-availability/coordinating-only-dedicated-coordinating-node/

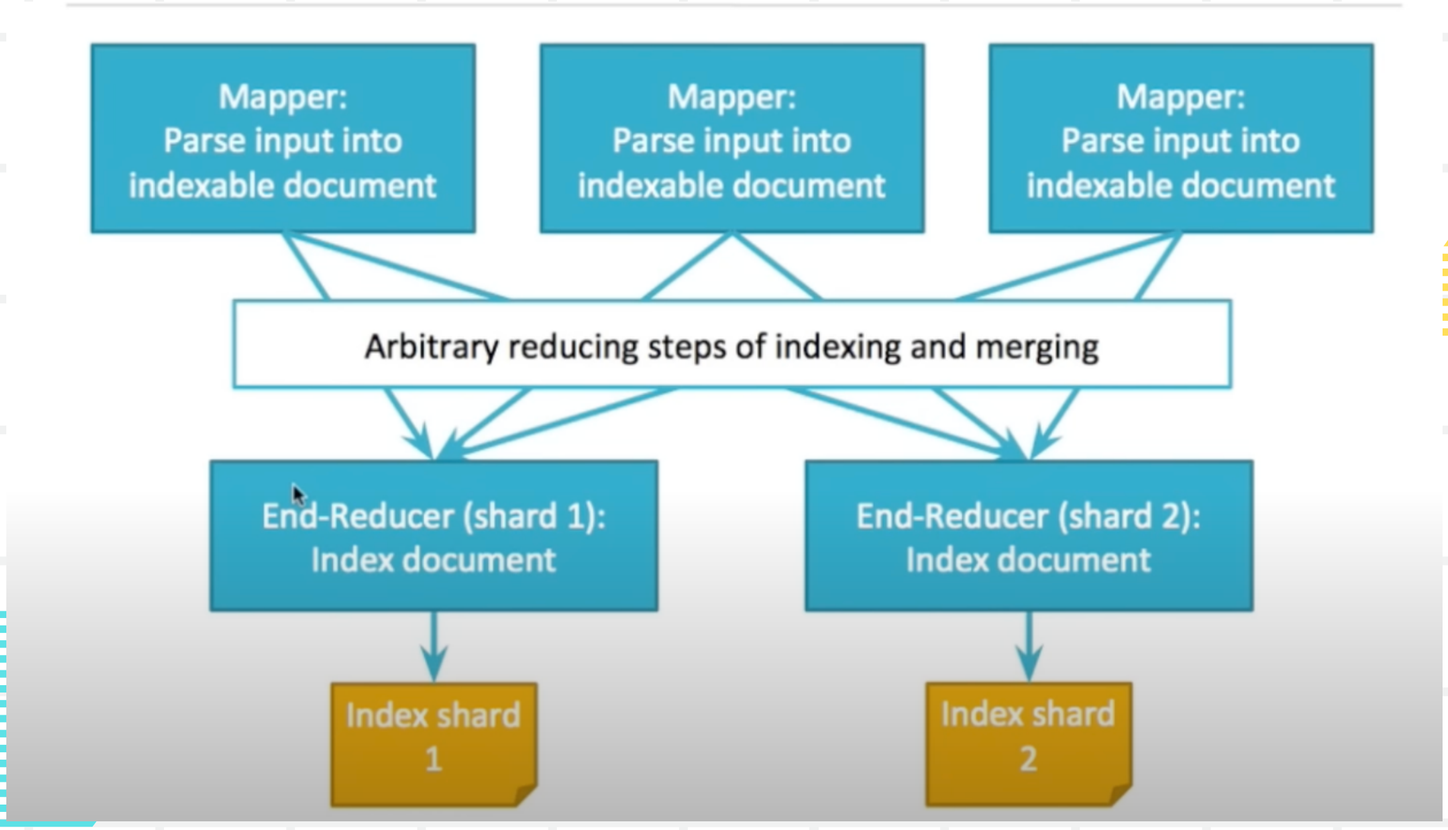
DISTRIBUTED CONCEPTS IN ELASTICSEARCH



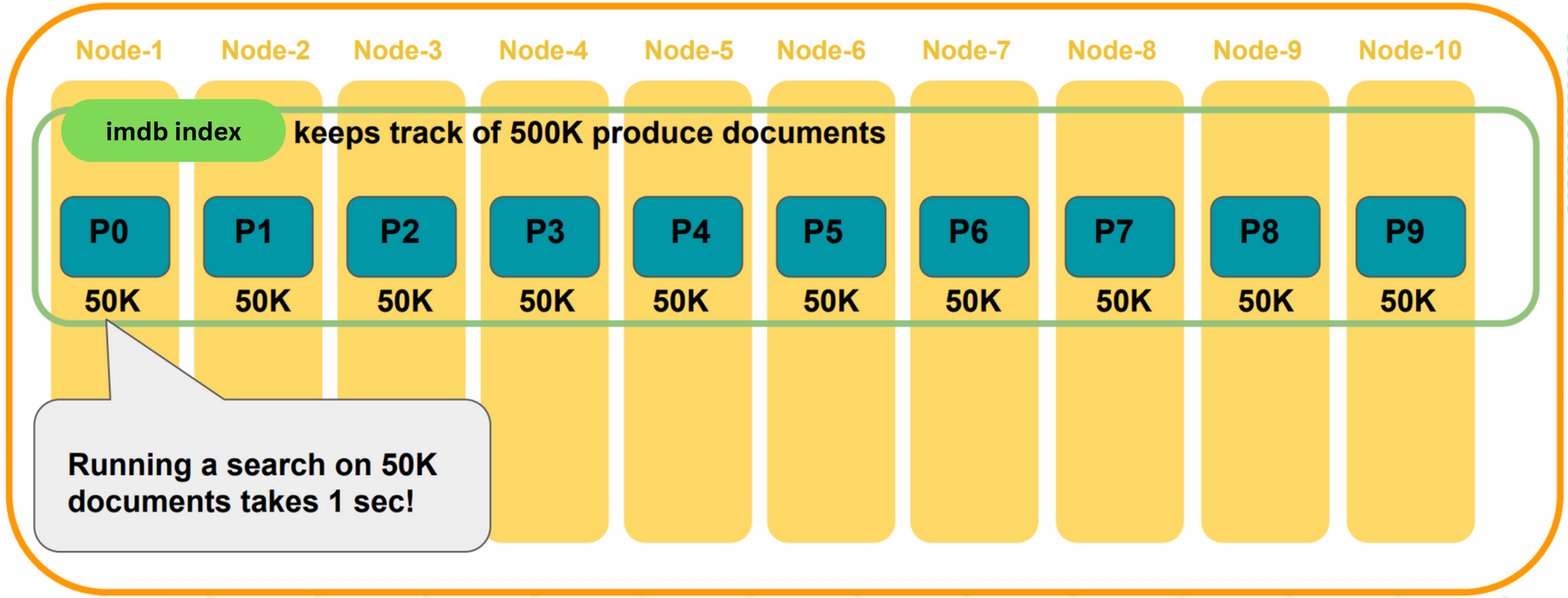
https://www.elastic.co/blog/how-many-shards-should-i-have-in-my-elasticsearch-cluster

https://www.elastic.co/guide/en/elasticsearch/guide/current/inside-a-shard.html

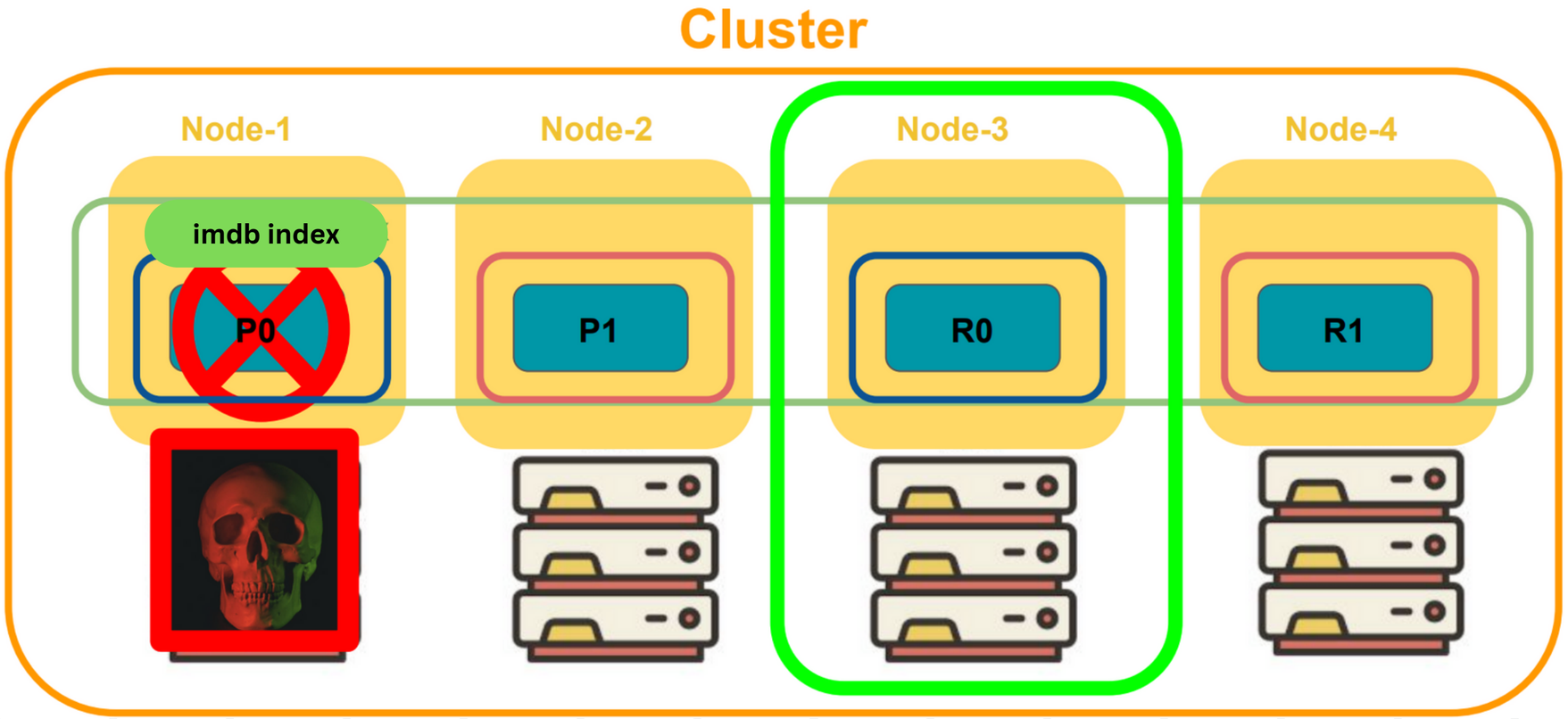
DISTRIBUTED CONCEPTS IN ELASTICSEARCH



DISTRIBUTED CONCEPTS IN ELASTICSEARCH

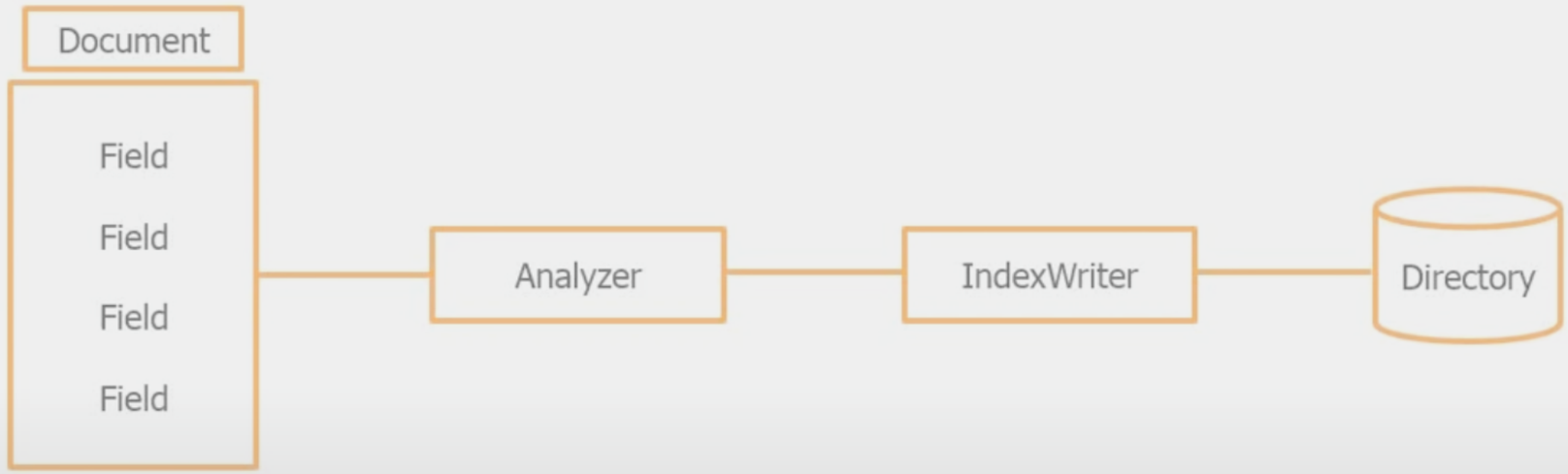


DISTRIBUTED CONCEPTS IN ELASTICSEARCH



Are replicas only useful for fault tolerance?

DOCUMENT INDEXING



Importance of Indexing:

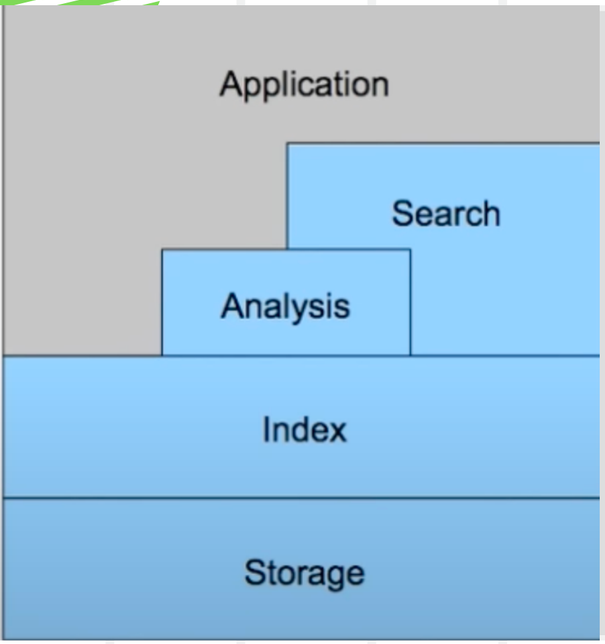
- Facilitates fast and accurate search by organizing and structuring data.
- Improves search relevance and performance by enabling efficient query processing.
- Enables features such as full-text search, aggregations, and sorting.

DOCUMENT INDEXING

Steps in Document Indexing:

- **Tokenization**:
  - The process of breaking text into individual tokens or terms.
  - Tokens are the smallest units of searchable text.
- **Analysis**:
  - Tokenized text undergoes analysis, which involves applying filters and analyzers to transform tokens into standardized forms.
  - Analyzers include stemming, lowercase conversion, and removal of stop words.
- **Mapping**:
  - Defines the structure of the index by specifying data types and field properties.
  - Determines how fields are analyzed and indexed.
- **Indexing**:
  - Process of storing analyzed and transformed documents in the index.
  - Indexed documents become searchable and retrievable.

APACHE LUCENE



Apache Lucene:

- Apache Lucene is a high-performance, full-featured text search engine library written in Java.
- It provides powerful indexing and search capabilities used by various applications and frameworks.

Role in Elasticsearch:

- Elasticsearch is built on top of Apache Lucene, leveraging its core capabilities for indexing and search.
- Lucene serves as the underlying search engine powering Elasticsearch's distributed search and analytics functionalities.

Indexing with Lucene:

- Lucene provides robust indexing capabilities for organizing and storing textual data efficiently.
- It employs inverted indexing to map terms to document IDs, enabling fast retrieval of documents matching a given query.

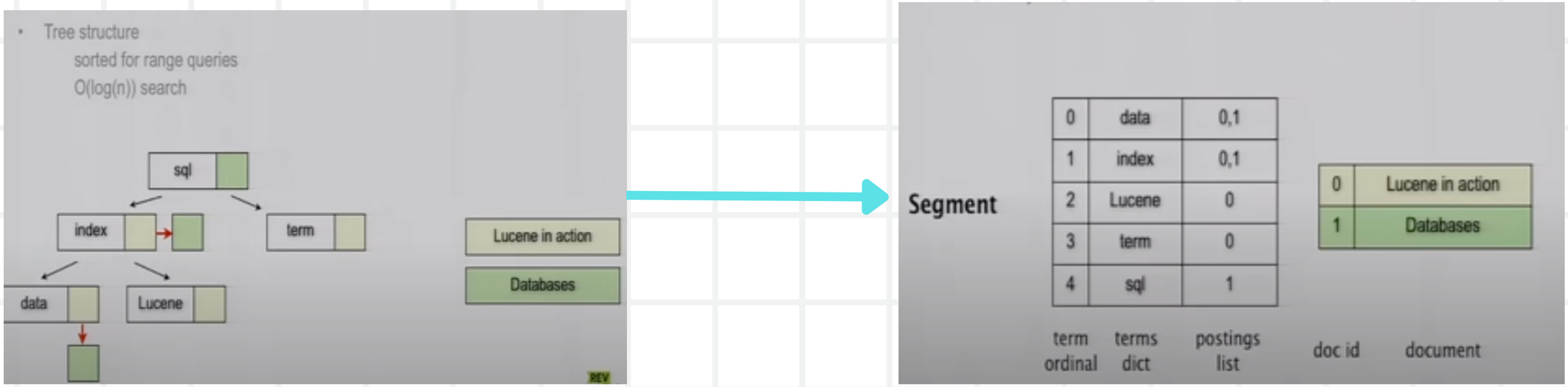
Searching with Lucene:

- Lucene offers advanced searching features, including support for Boolean queries, phrase queries, wildcard queries, and more.
- It utilizes data structures like inverted indexes and data structures like tries to achieve efficient search operations.

Elasticsearch Integration:

- Elasticsearch extends Lucene's capabilities with additional features like distributed search, real-time indexing, and RESTful APIs.
- It provides a user-friendly interface and scalability features while retaining the robustness of Lucene's search capabilities.

APACHE LUCENE



APACHE LUCENE

Insertions?

- Insertion = write a new segment
- Merge segments when there are too many of them
  concatenate docs, merge terms dicts and postings lists (merge sort!)

0	data	0
1	index	0
2	Lucene	0
3	term	0

0	Lucene in action

0	data	0
1	index	0
2	sql	0

0	Databases

0	data	0,1
1	index	0,1
2	Lucene	0
3	term	0
4	sql	1

0	Lucene in action
1	Databases

Deletions?

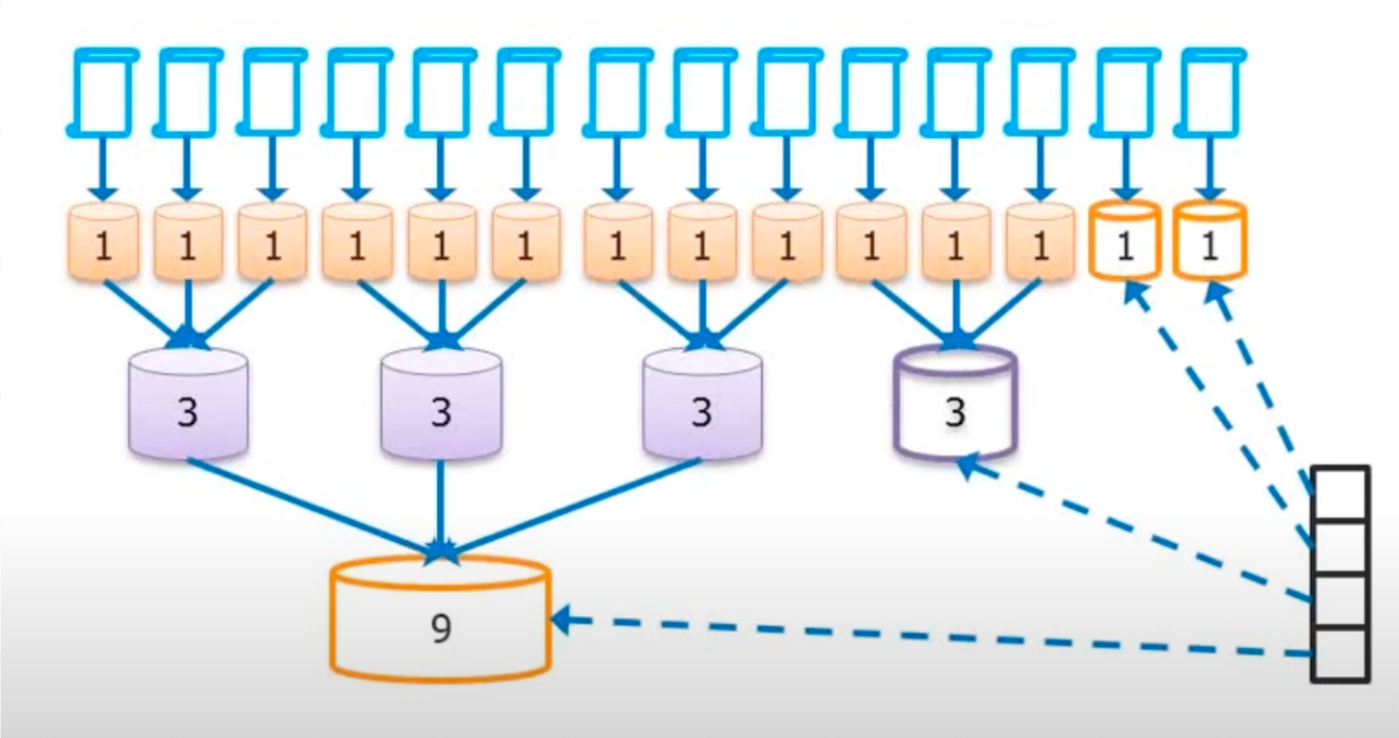
- Deletion = turn a bit off
- Ignore deleted documents when searching and merging (reclaims space)
- Merge policies favor segments with many deletions

0	data	0,1
1	index	0,1
2	Lucene	0
3	term	0
4	sql	1

0	Lucene in action	1
1	Databases	0

live docs: 1 = live, 0 = deleted

APACHE LUCENE



APACHE LUCENE

- Updates require writing a new segment
  - single-doc updates are costly, bulk updates preferred
  - **writes are sequential**
- Segments are never modified in place
  - filesystem-cache-friendly
  - **lock-free!**

index is incrementally updateable
- newly added documents go in tiny new indexes
- bulk of index is in larger index segments
- segments merged in background
- lazy merge sort: optimal, scalable, n*log(n)

supports fast search
- search all segments
- worst case: $k \times \log_k(n)$ seeks per query term
- optimized index: single seek per term

With no changing segments, it would be very good for the file system to cache the data and not to read from the disk.
By having immutable data, we have a lock-free mechanism.

APACHE LUCENE

- Updates require writing a new segment
  - single-doc updates are costly, bulk updates preferred
  - **writes are sequential**
- Segments are never modified in place
  - filesystem-cache-friendly
  - **lock-free!**

index is incrementally updateable
- newly added documents go in tiny new indexes
- bulk of index is in larger index segments
- segments merged in background
- lazy merge sort: optimal, scalable, n*log(n)

supports fast search
- search all segments
- worst case: $k \times \log_k(n)$ seeks per query term
- optimized index: single seek per term

With no changing segments, it would be very good for the file system to cache the data and not to read from the disk.
By having immutable data, we have a lock-free mechanism.

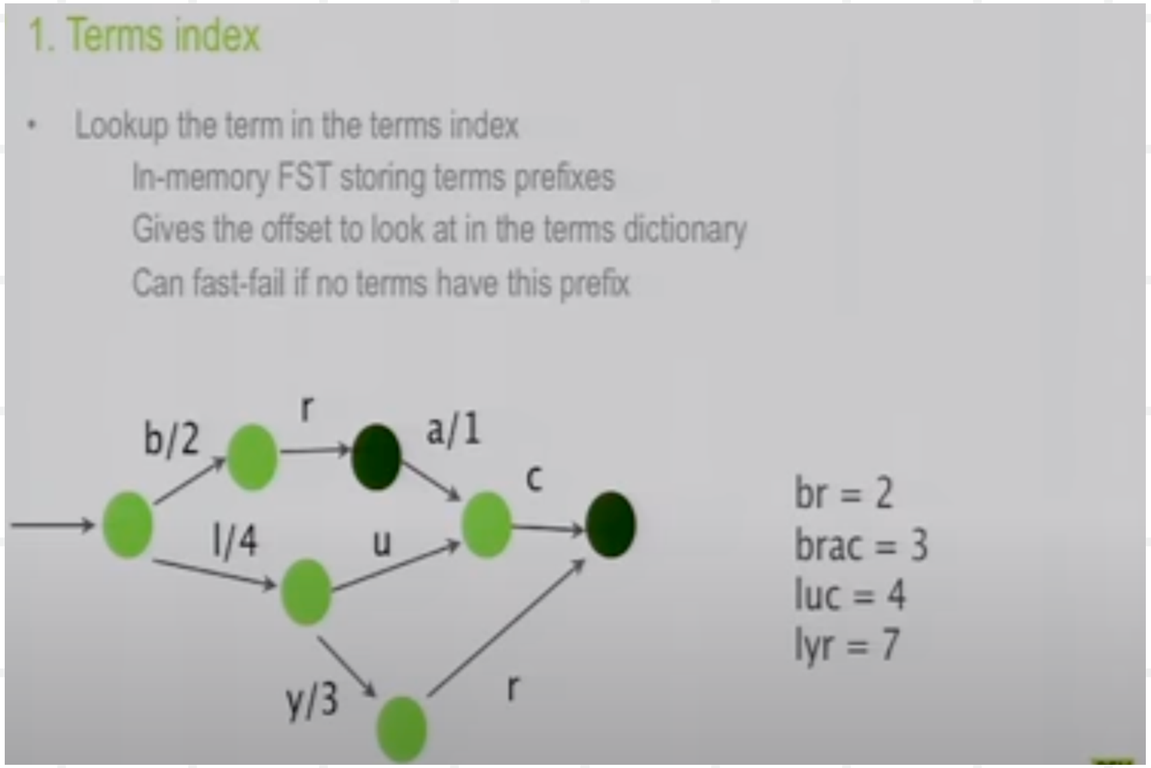
APACHE LUCENE

- **Using a compound file for the first segments.**
- **The smaller, the better for caching.**
- **Using Codec for file storage.**
- **We don't want the compression to be such a heavy process that the CPU becomes a bottleneck.**

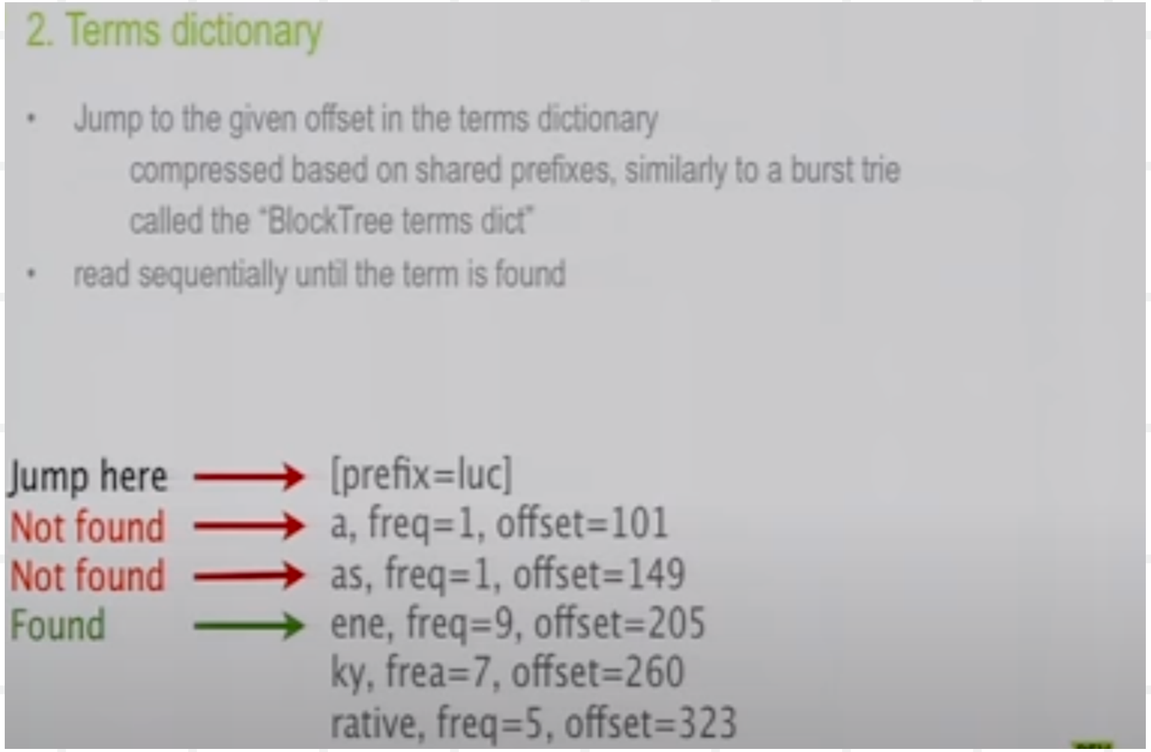
APACHE LUCENE

What happens when you run a query?

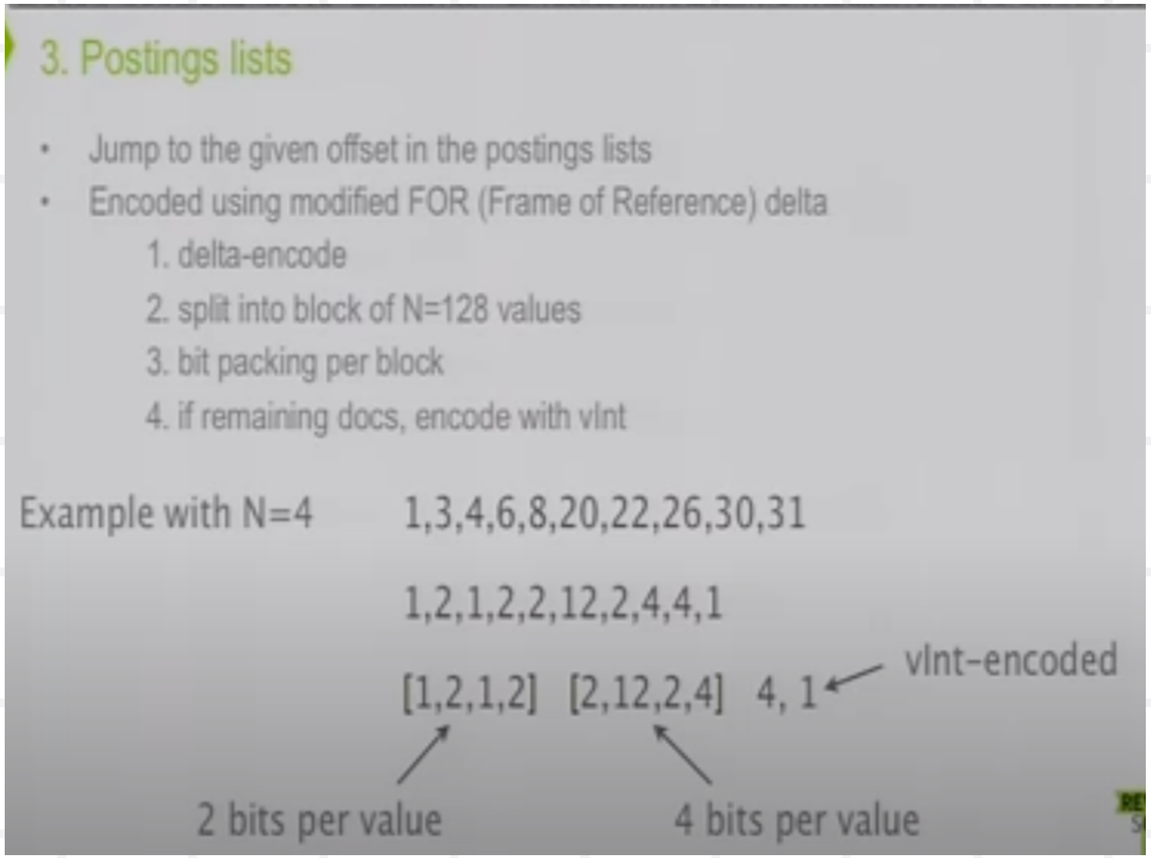
APACHE LUCENE



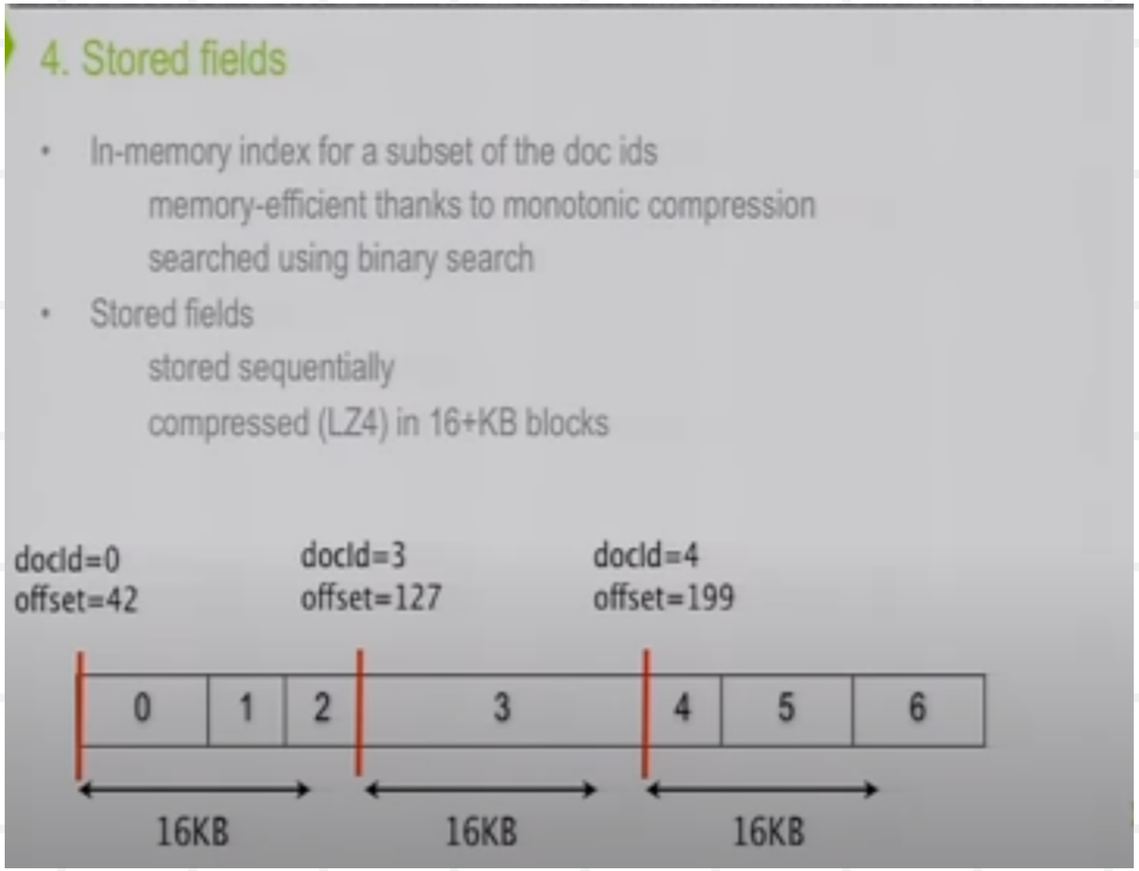
APACHE LUCENE



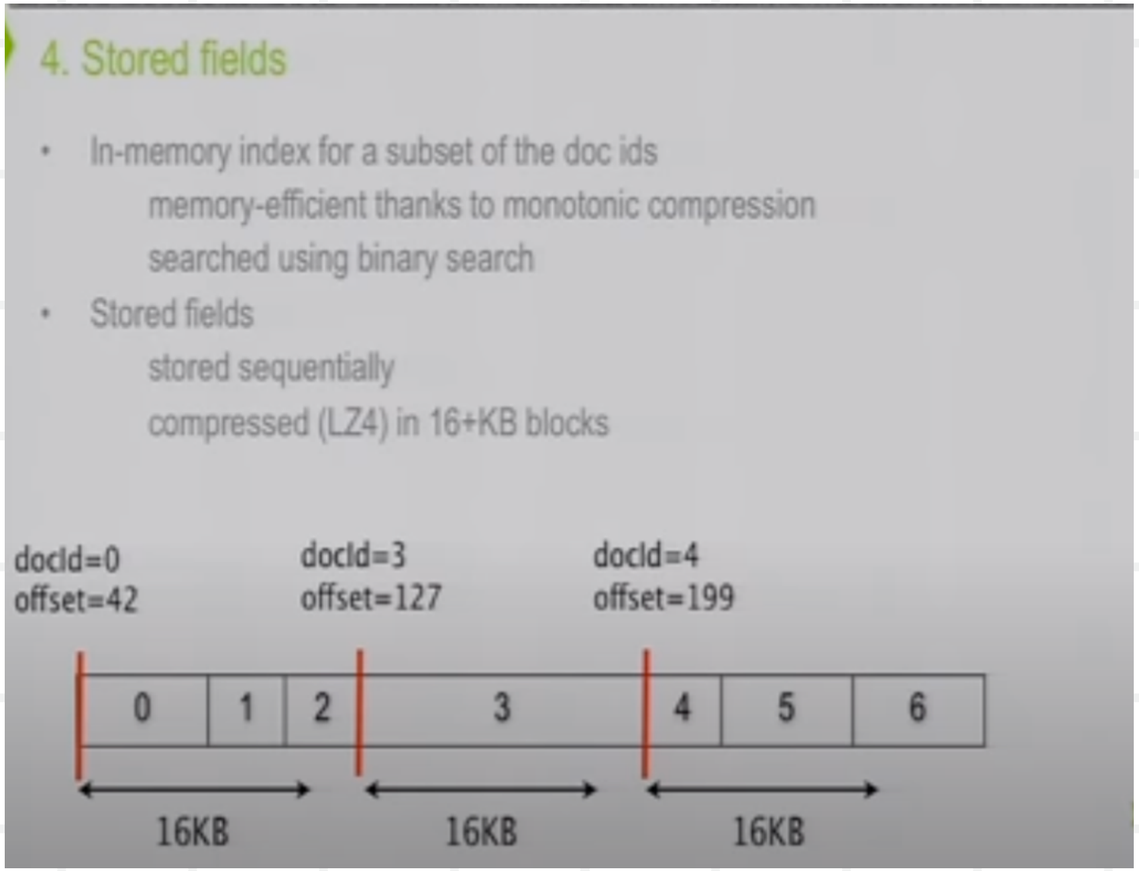
APACHE LUCENE



APACHE LUCENE



APACHE LUCENE



APACHE LUCENE

Query execution

- 2 disk seeks per field for search
- 1 disk seek per doc for stored fields

- It is common that the terms dict / postings lists fits into the file-system cache

- "Pulse" optimization
  - For unique terms (freq=1), postings are inlined in the terms dict
  - Only 1 disk seek
  - Will always be used for your primary keys

APACHE LUCENE SCORER

```
Weight w = query.createWeight(searcher, true, 1.0);
for (LeafReaderContext ctx: reader.leaves()) {
    Scorer s = w.scorer(ctx);
    LeafCollector c = collector.getLeafCollector(ctx);
    c.setScorer(s);
    DocIdSetIterator it = s.iterator();
    while (it.nextDoc() != NO_MORE_DOCUMENTS) {
        c.collect(it.docID());
     }
}
```

- **The scoring is documented one at a time.**
- **After the first scoring, then rescoring happens. It will score with a more complicated algorithm like BM-25.**

REFERENCES

[1]: https://www.youtube.com/watch?v=gS_nHTWZEJ8&list=PL_mJOmq4zsHZYAyK606y7wjQtC0aoE6Es&index=1
[2]: https://github.com/LisaHJung/Beginners-Crash-Course-to-Elastic-Stack-Series-Table-of-Contents?tab=readme-ov-file
[3]: https://www.youtube.com/watch?v=lWKEphKIG8U
[4]: https://medium.com/geekculture/elasticsearch-internals-4c4c9ec077fa
[5]: https://assets.ctfassets.net/oxjq45e8ilak/4abnyTszsKaBhPWDMpLYsu/31244b4286327838faf42c6b77b87843/Elasticsearch_internals.pdf
[6]: https://www.elastic.co/blog/found-elasticsearch-from-the-bottom-up
[7]: https://buildatscale.tech/how-elasticsearch-works-internally/
[8]: https://www.youtube.com/watch?v=T5RmMNDR5XI
[9]:https://www.youtube.com/watch?v=YsYUgZu9-Y4
[10]:https://github.com/elastic/elasticsearch
[11]:https://github.com/apache/lucene/blob/main/lucene/core/src/java/org/apache/lucene/index/IndexWriter.java
[12]:https://people.apache.org/~mikemccand/lucenebench/
[13]:https://en.wikipedia.org/wiki/Compound_File_Binary_Format
[14]:https://lucene.apache.org/core/9_10_0/core/org/apache/lucene/index/package-summary.html#writer
[15]:https://www.youtube.com/watch?v=aG6WPu08qBs
[16]:https://www.linkedin.com/pulse/how-does-elasticsearch-handle-distributed-search-indexing-singh/
[17]:https://opster.com/guides/elasticsearch/high-availability/coordinating-only-dedicated-coordinating-node/

THANK YOU

Presentation by Amir Hossein Rahmati

Modern Information Retrieval Course lectured by Dr. Soleymani | 2024 | Sharif University of Technology