



Elasticsearch Search Improvements

Nicholas Knize, Lee Hinman, Jim Ferenczi

Elastic

2017-03-09

@nkinze @thnetos @jimferenczi

Agenda

1

Range Fields

2

Removing the `_all` field

3

Unified Highlighter

4

Multi-token Synonyms

Range Fields












Range Fields

And why is it useful

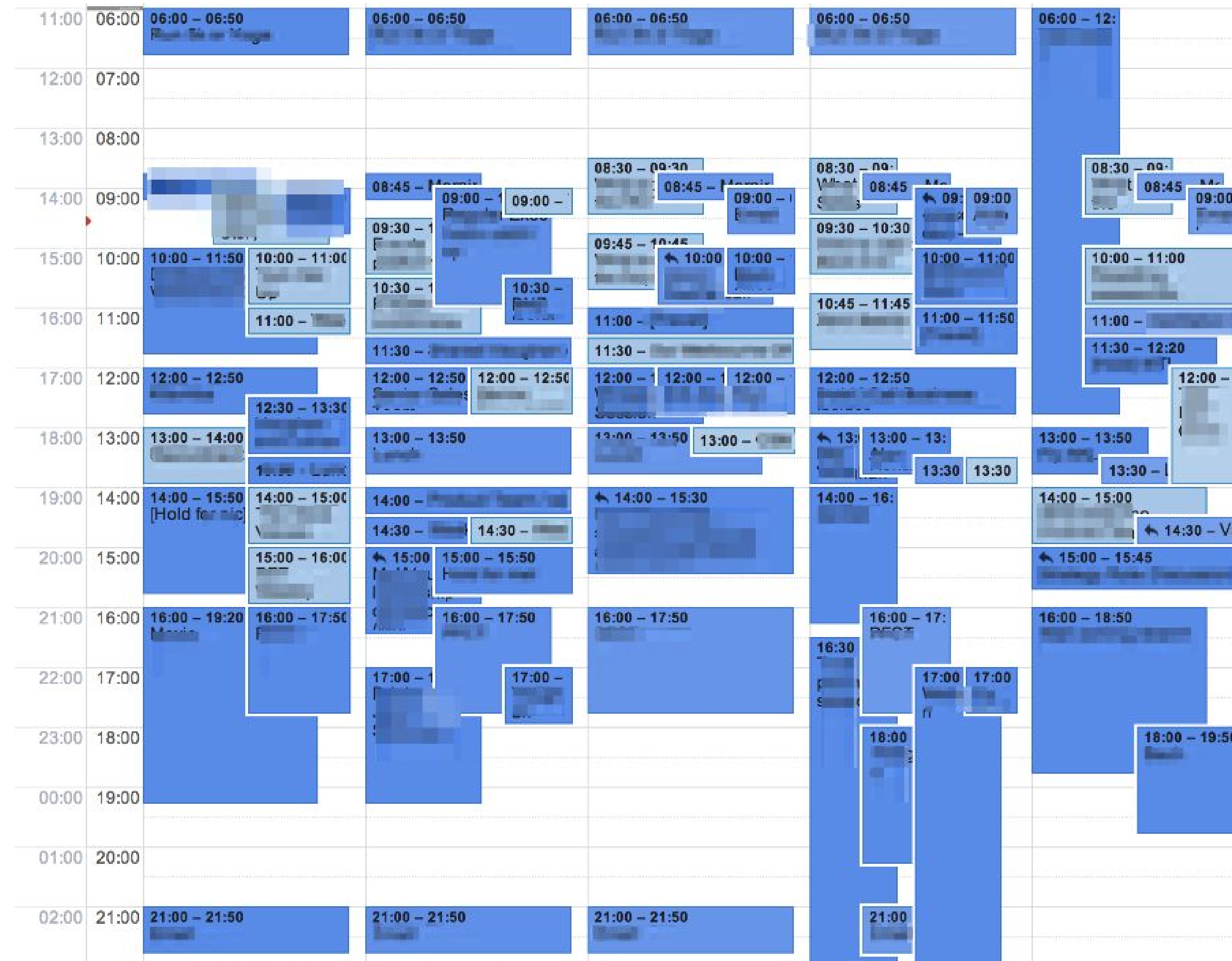
Range Fields provide the ability to index a continuous series of values (e.g., real numbers)

- `from` and `to` values are handled for you and optimized for POINTS (Bkd) data structure
- range queries find all documents whose range field relate to a desired range (e.g., `WITHIN`, `INTERSECTS`)
- Lays the foundation for multi-dimensional ranges (e.g., Bounding Boxes, Cubes)

TV Guide Search

	8:00PM	8:30PM	9:00PM	9:30PM
	The Big Bang Theory NEW 8PM - 8:31PM ★		The Big Bang Theory 8:31PM - 9PM ★	
	Gotham NEW 8PM - 9PM ★		Sleepy Hollow NEW 9PM - 10PM ★	
	The Voice NEW 8PM - 10PM			
	The Originals NEW 8PM - 9PM		Jane the Virgin NEW 9PM - 10PM	
	Duck Commander: Before the Dynasty 8PM - 8:30PM	Duck Commander: Before the Dynasty 8:30PM - 9PM	Duck Commander: Before the Dynasty 9PM - 9:30PM	Duck Commander: Before the Dynast 9:30PM - 10:01PM
	Harry Potter and the Sorcerer's Stone 7:30PM - 11PM			
	The Bucket List 8PM - 10:01PM			
	Top Gear 8PM - 9PM		Top Gear NEW 9PM - 10PM	
	ATL 7PM - 9:30PM			Paid in Full 9:30PM - 12AM
	The Knick 8PM - 8:45PM		The Wolverine 8:45PM - 11PM	
	Shark Tank 8PM - 9PM		Shark Tank 9PM - 10PM	

Event Search



Numeric Field Types

Discrete Values

- integer
- float
- long
- double
- date
- short
- half_float
- scaled_float

Range Field Types

Continuous Values

- integer
 - float
 - long
 - double
 - date
 - short
 - half_float
 - scaled_float
- **integer_range**
 - **float_range**
 - **long_range**
 - **double_range**
 - **date_range**

Range Field Mappings

Define

```
PUT events/conference/_mapping
{
  "properties" : {
    "name" : {
      "type" : "text"
    },
    "expected_attendees" : {
      "type" : "integer_range",
    },
    "time" : {
      "type" : "date_range",
      "format" : "yyyy-MM-dd HH:mm:ss||epoch_millis",
    }
  }
}
```

Range Fields

Insert

```
PUT events/conference/1
{
  "name" : "elasticon",
  "expected_attendees" : {
    "gte" : 5000,
    "lte" : 10000
  },
  "time" : {
    "gte" : "2017-03-07 9:00",
    "lte" : "2017-03-09 17:00"
  }
}
```

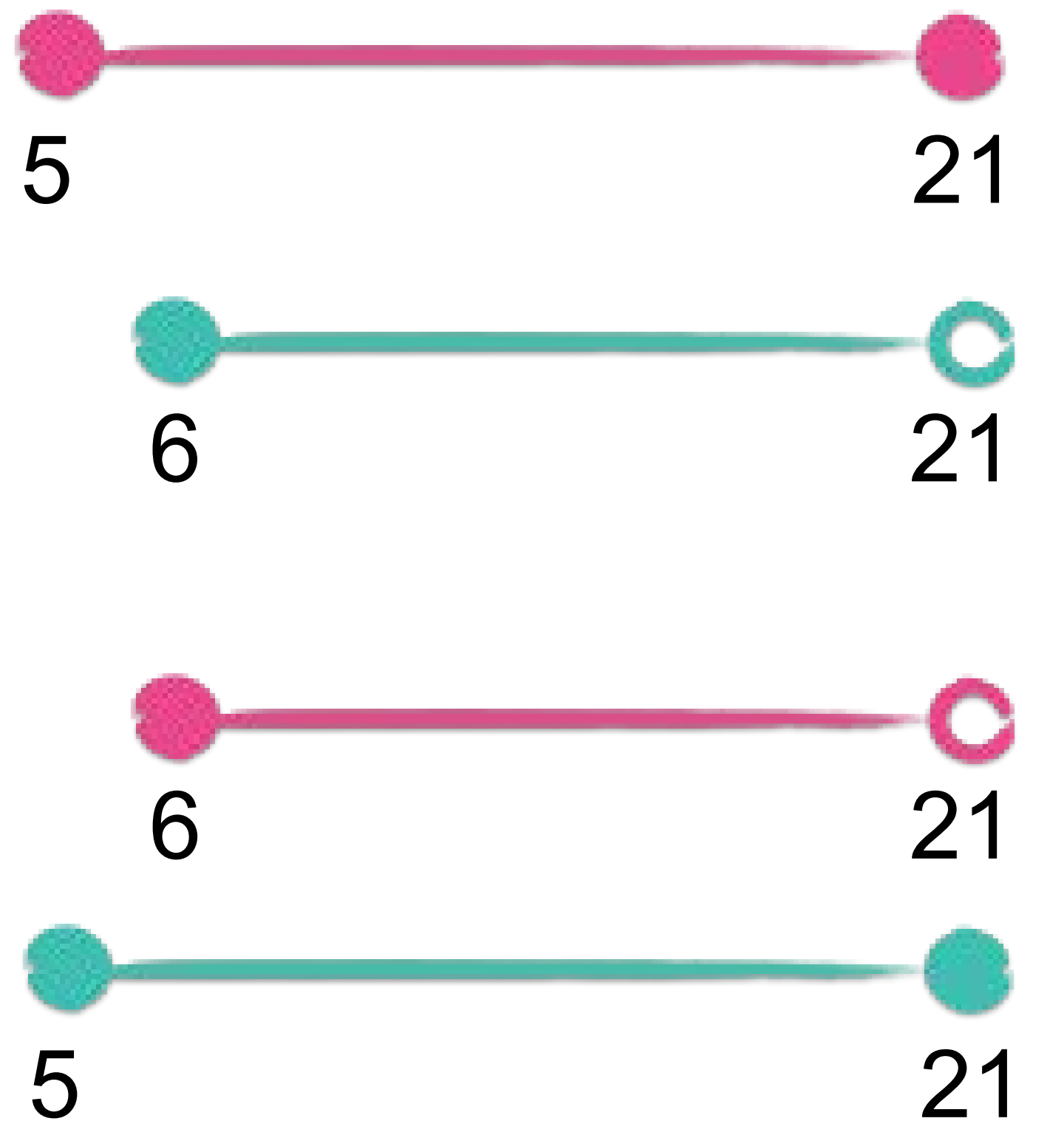
Range Fields

Query

```
GET events/_search
{
  "query" : {
    "range" : {
      "time" : {
        "gte" : "2017-03-05",
        "lte" : "2017-03-10",
        "relation" : "within"
      }
    }
  }
}
```

Range Fields

Relations



***INTERSECTS,
WITHIN***

***INTERSECTS,
CONTAINS***



INTERSECTS

Removing _all

What does `_all` provide?

And why is it useful

The `_all` field provides the ability to search without knowing anything about mappings

- The `query_string` and `simple_query_string` queries both search `_all` by default
- Allows construction of a single text box like Kibana where queries can be “free-form”
- No need to worry about field types not being handled correctly, everything is treated as a string

What's wrong with `_all`?

Why remove it?

The `_all` field has a number of shortcomings

- Data is duplicated in `_all` and your other fields
- Numeric data does not compress well since it is interpreted as a string
- `_all` has only one analyzer, and does not use the per-field analysis when querying
- Highlighting “gotchas” due to `_all` not being a real field

The Best of Both Worlds

Keeps the pros and ditch the cons

We've added an `all_fields` mode to the `query_string` and `simple_query_string` queries

For every field that can be automatically queried, we add that field to the `fields` parameter and leniently parse the query text, ignoring text that cannot be parsed for the field (such as plain text on numeric fields).

Doing it automatically

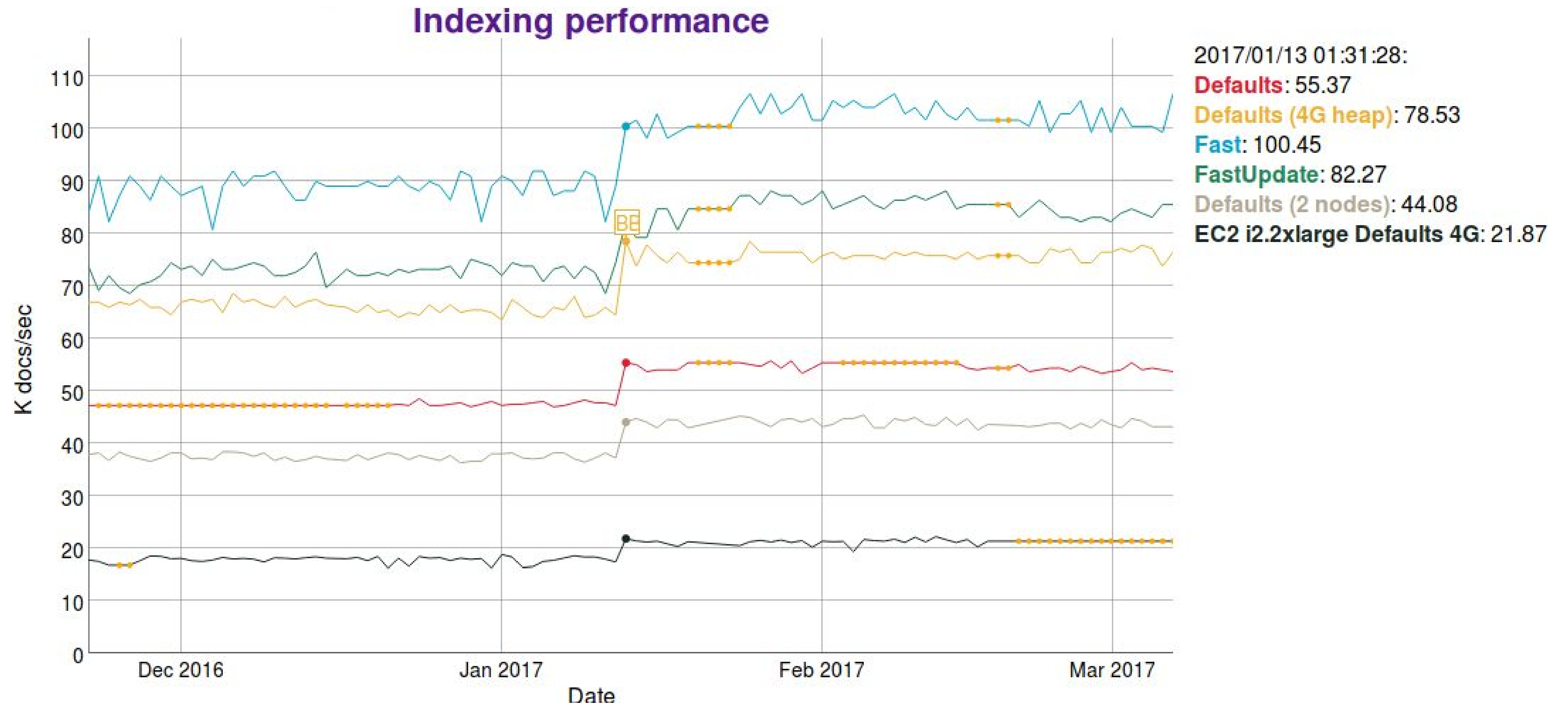
Making it happen without intervention

The `all_fields` mode kicks in automatically when the following criteria are met, in Elasticsearch 5.1.1 or later

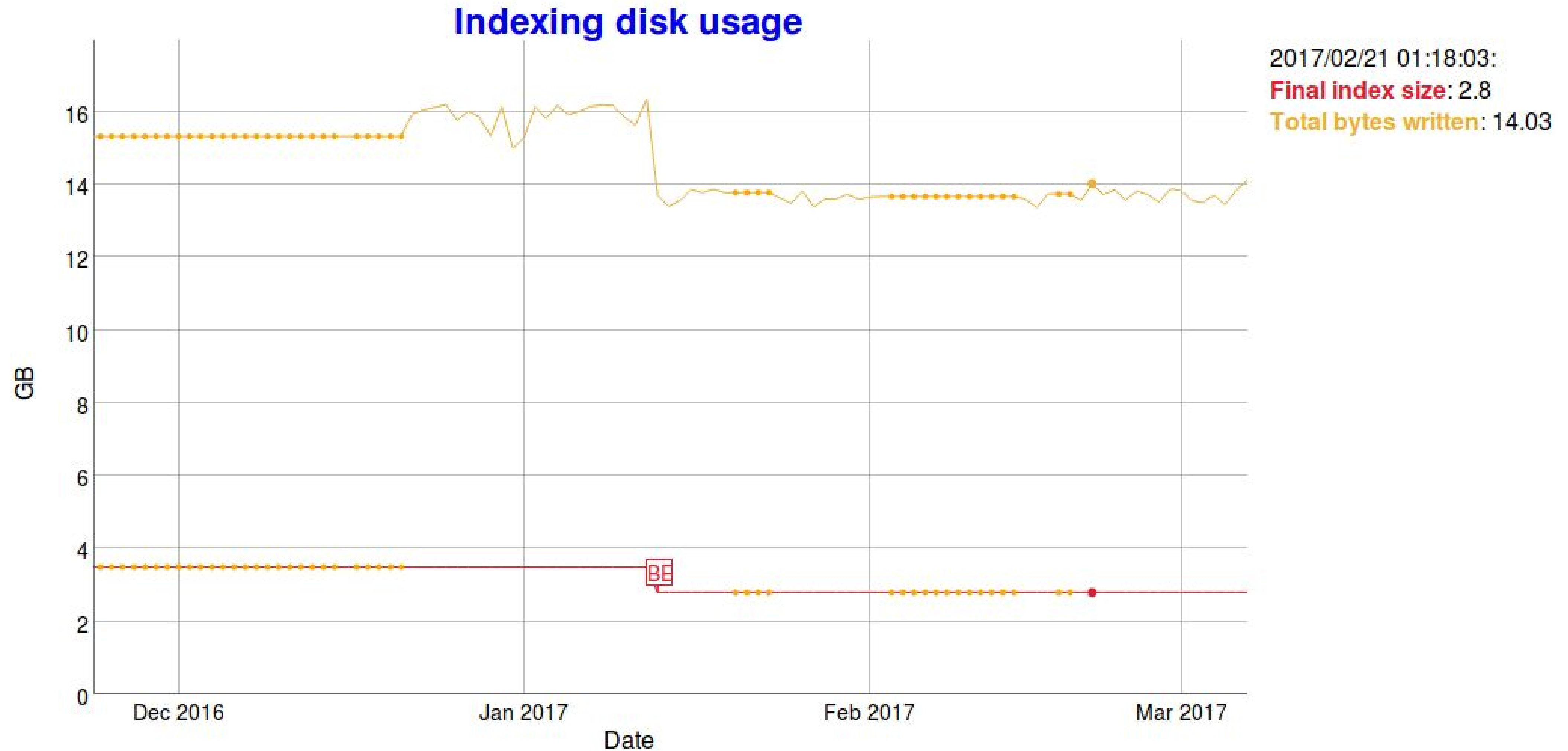
- The `_all` field is disabled
- No `default_field` has been set in the index settings (`index.query.default_field`)
- No `default_field` has been set in the request
- No `fields` are already specified in the request

With this, `_all` will be disabled by default and not configurable in Elasticsearch 6.0.

Benchmarks without _all



Benchmarks without _all



Manually force all_fields execution mode

If you want to try it with `_all` still enabled

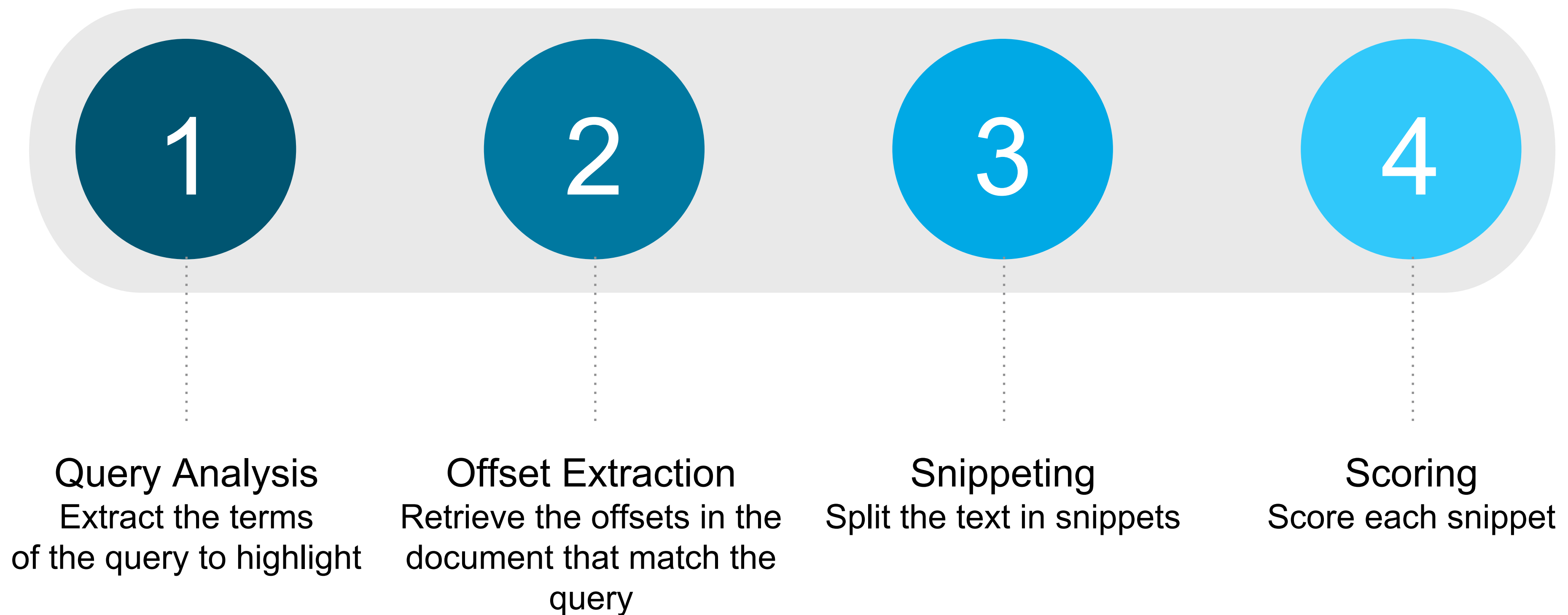
```
POST /_search
{
  "query": {
    "query_string": {
      "query": "secarity~1 f_int:200 127.0.0.1 \"2016/09/01\\\"\",
      "default_operator": "AND",
      "all_fields": true
    }
  }
}
```

Unified Highlighter

Highlighting

- Part of the search experience
- Why a document matched the query
- Visual clue for textual fields
- Full content highlighting
- Summarize content centered on the user query
 - Extracts the best snippets (passages) of the document that matched the query (partially or not)

Highlighting Process



Highlighters

Under the scene ES uses 3 different highlighters:

- Plain
 - Re-analyze the text
- FVH
 - Leverage the term vectors to extract the offsets
- Postings
 - Extract offsets from the postings lists

Why a new Highlighter

- Strongly typed
- Different experience
 - Query analysis
 - Snippetting
 - Scoring
- Difficult to maintain
 - Different feature set
 - Code is not shared

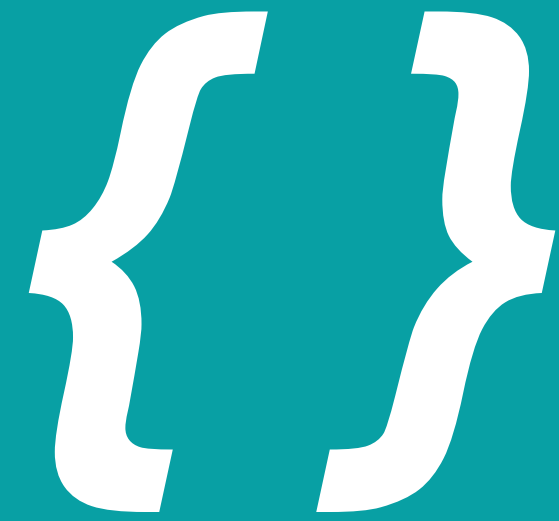
Unified Highlighter

- Lucene 6.4
- Agnostic to types
- Adaptive Offsets Extraction:
 - Field-dependent
 - Query-dependent
- Fork of the PostingsHighlighter
- Merge good practice from other types

Unified Highlighter

- Based on Spans:
 - Handle complex positional queries
- Handle multi-term queries (prefix, wildcard, ...)
- Designed for speed and relevancy
- Scoring model based on BM25:
 - Treats the document as the whole corpus, and scores individual sentences as if they were documents in this corpus, using the BM25 algorithm
- Experimental

Multi-token Synonyms



*A longstanding, complex, glaring
and embarrassing limitation of
synonyms with Lucene*

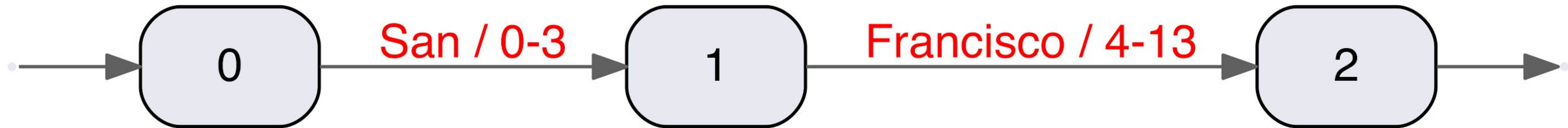
Michael McCandless

Analyzer and TokenStream

- Text Analysis
 - Indexation
 - Query
- Produces a sequence of tokens with specific attributes
- Positions in a TokenStream are represented as increment:
 - Each token in the stream has a position increment ≥ 0

San Francisco

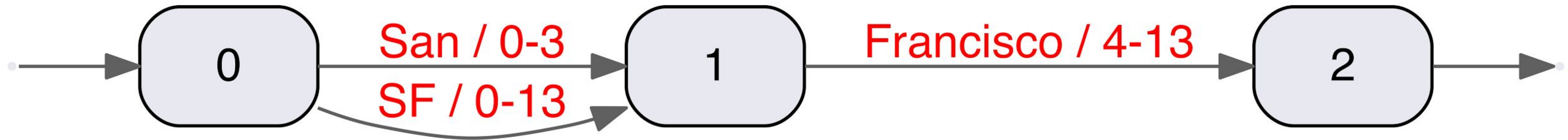
Lucene's TokenStream



San Francisco

SF

Lucene's TokenStream and Synonyms



San Francisco SF

Fog City

San Francisco
SF

Fog City

Frisko

San Francisco
SF

Fog City
Frisco

The City by the Bay

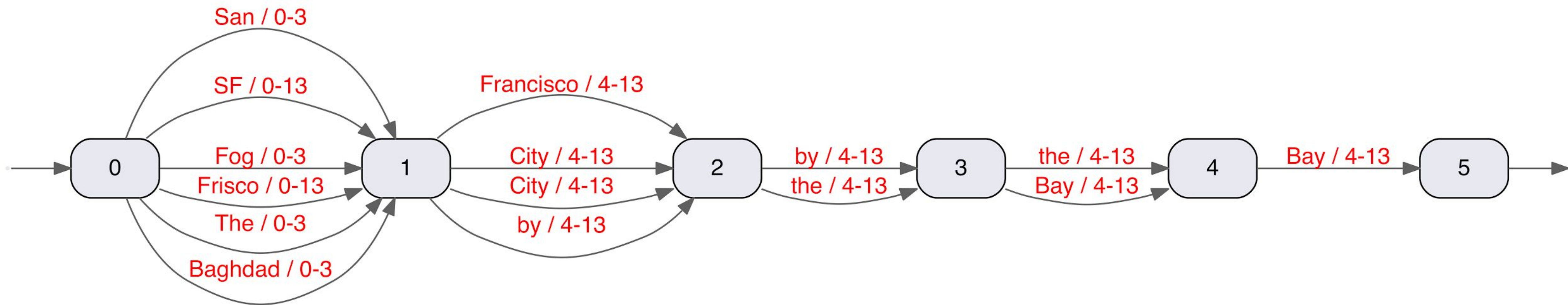
San Francisco
SF

Fog City
Frisco

The City by the Bay

Baghdad by the bay

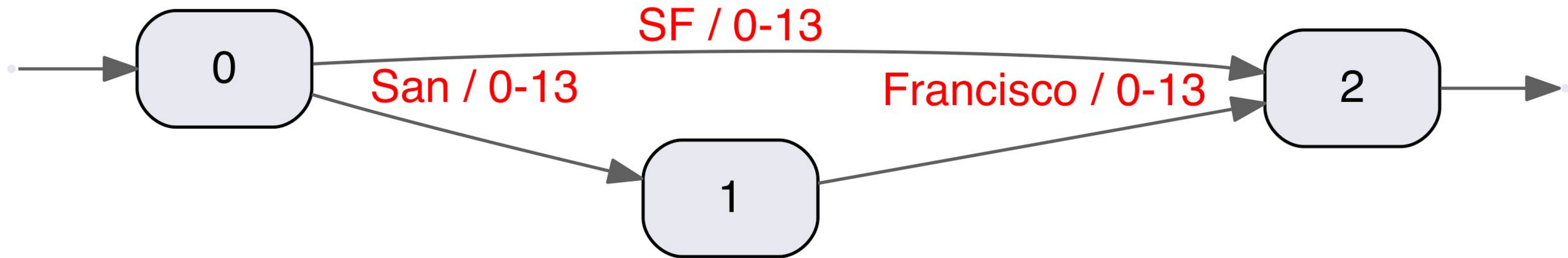
Lucene's TokenStreams and Synonyms



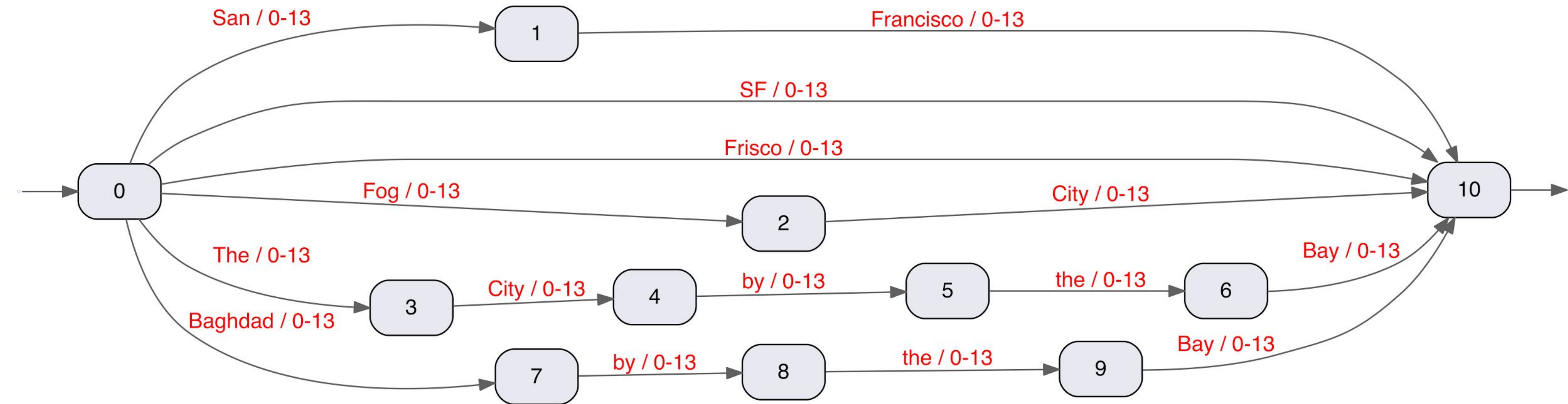
Position Length Attribute

- Determines how many positions a token spans
- Lucene's TokenStream are actually graphs !
- Graph Filters:
 - SynonymGraphFilter
 - ◉ Handle synonyms of different length (cf: SF, San Francisco)
 - ShingleFilter
 - JapaneseTokenizer
 - ◉ To express a whole word at the same time as possible sub-words
 - WordDelimiterGraph (ES 5.4)

Lucene's TokenStream are actually graphs !



Lucene's TokenStream are actually graphs !

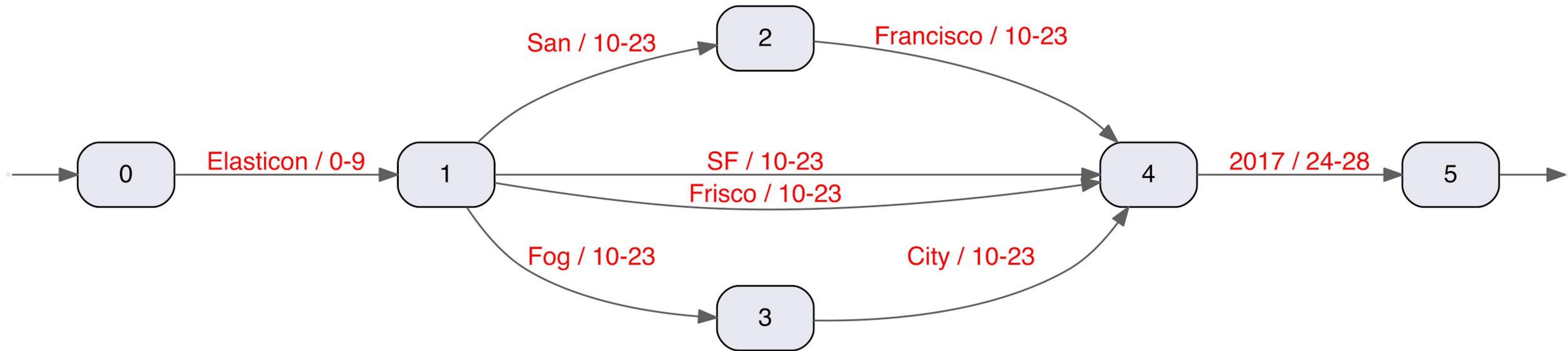


Search-time synonyms

- Query Parser “graph” aware:
 - Lucene QueryBuilder
- ES queries:
 - match, multi_match
 - query_string, simple_query_string (split_on_whitespace)
- Simple heuristic:
 - Enumerate all paths to build a query

Elasticon San Francisco 2017

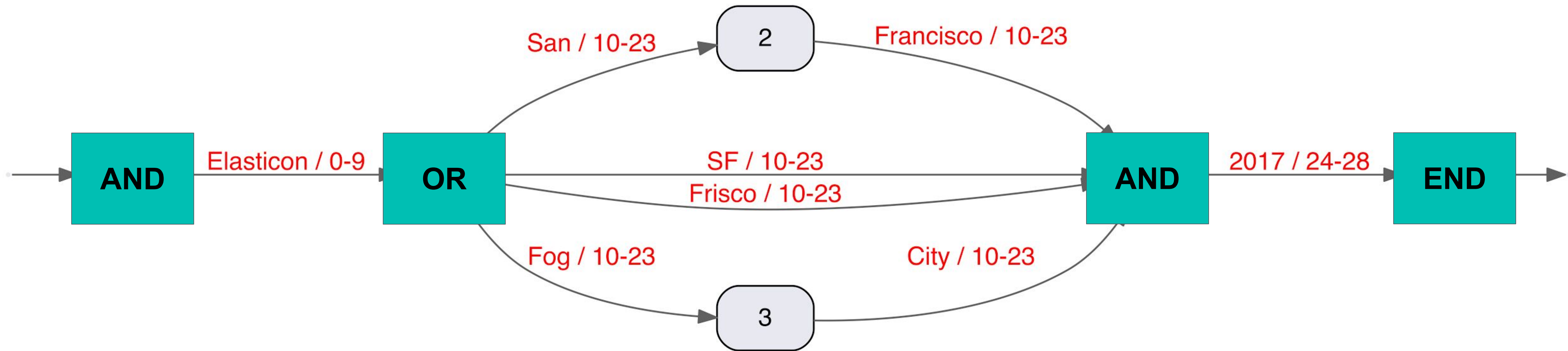
Query Builder



Query Builder

- Enumerate all paths:
 - (Elasticon San Francisco 2017) OR (Elasticon SF 2017) OR (Elasticon Fog City 2017)
- Can be inefficient:
 - Lots of duplicate terms
 - Combinatorial explosion with multiple synonyms in the same query
- Heuristic 2:
 - Find all articulation points in the graph and enumerate all path between them

Query Builder



More Questions?

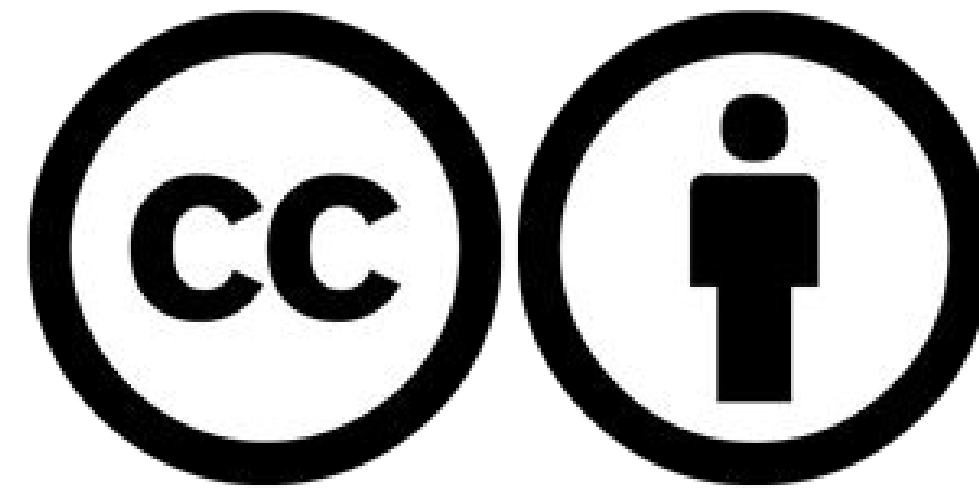
Visit us at the AMA



[www.elastic.c](http://www.elastic.co)

o

Please attribute Elastic with a link to elastic.co



Except where otherwise noted, this work is licensed under
<http://creativecommons.org/licenses/by-nd/4.0/>

Creative Commons and the double C in a circle are
registered trademarks of Creative Commons in the United States and other countries.
Third party marks and brands are the property of their respective holders.