Document relations

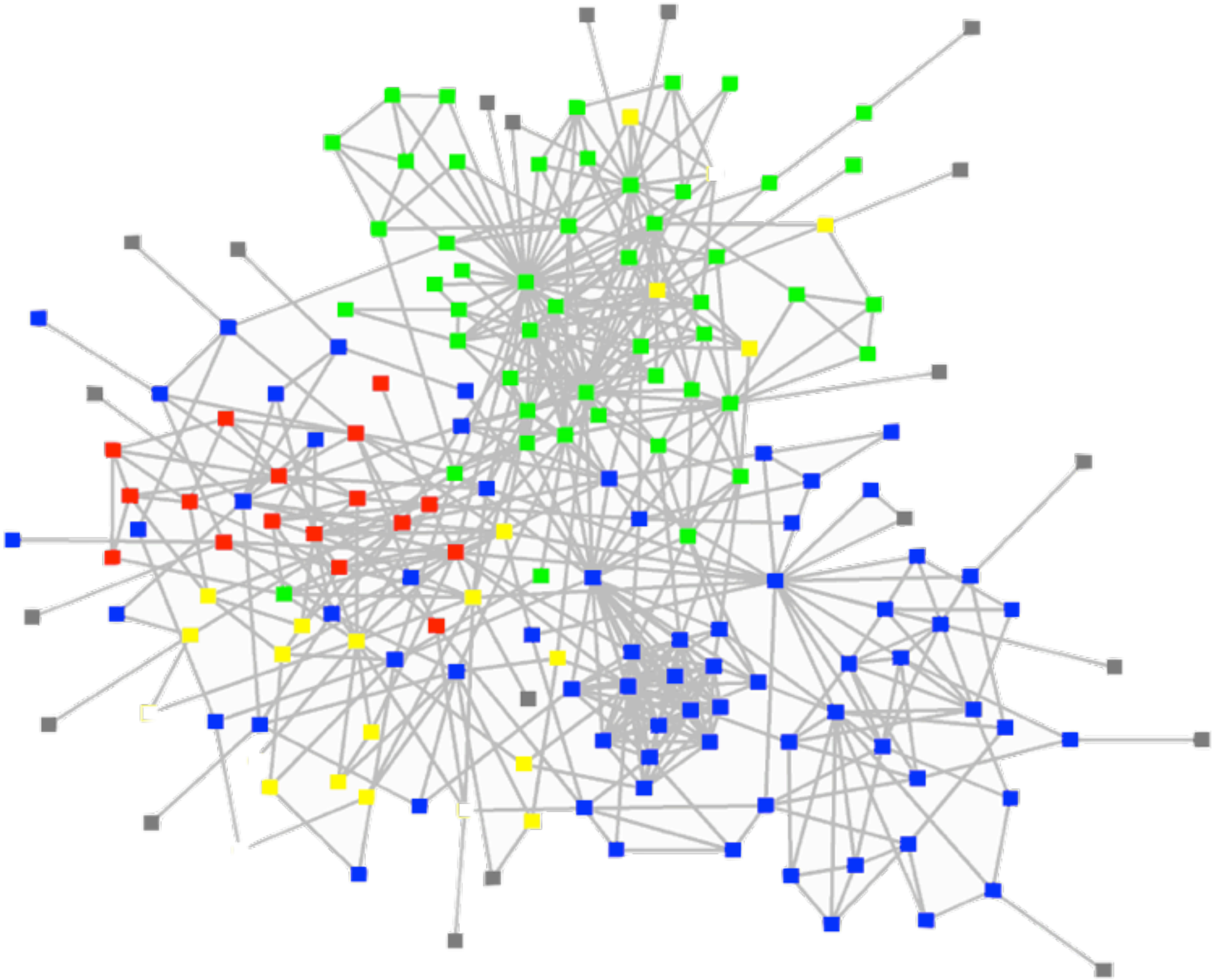
Martijn van Groningen
mvg@apache.org
@mvgroningen

elasticsearch.

Topics

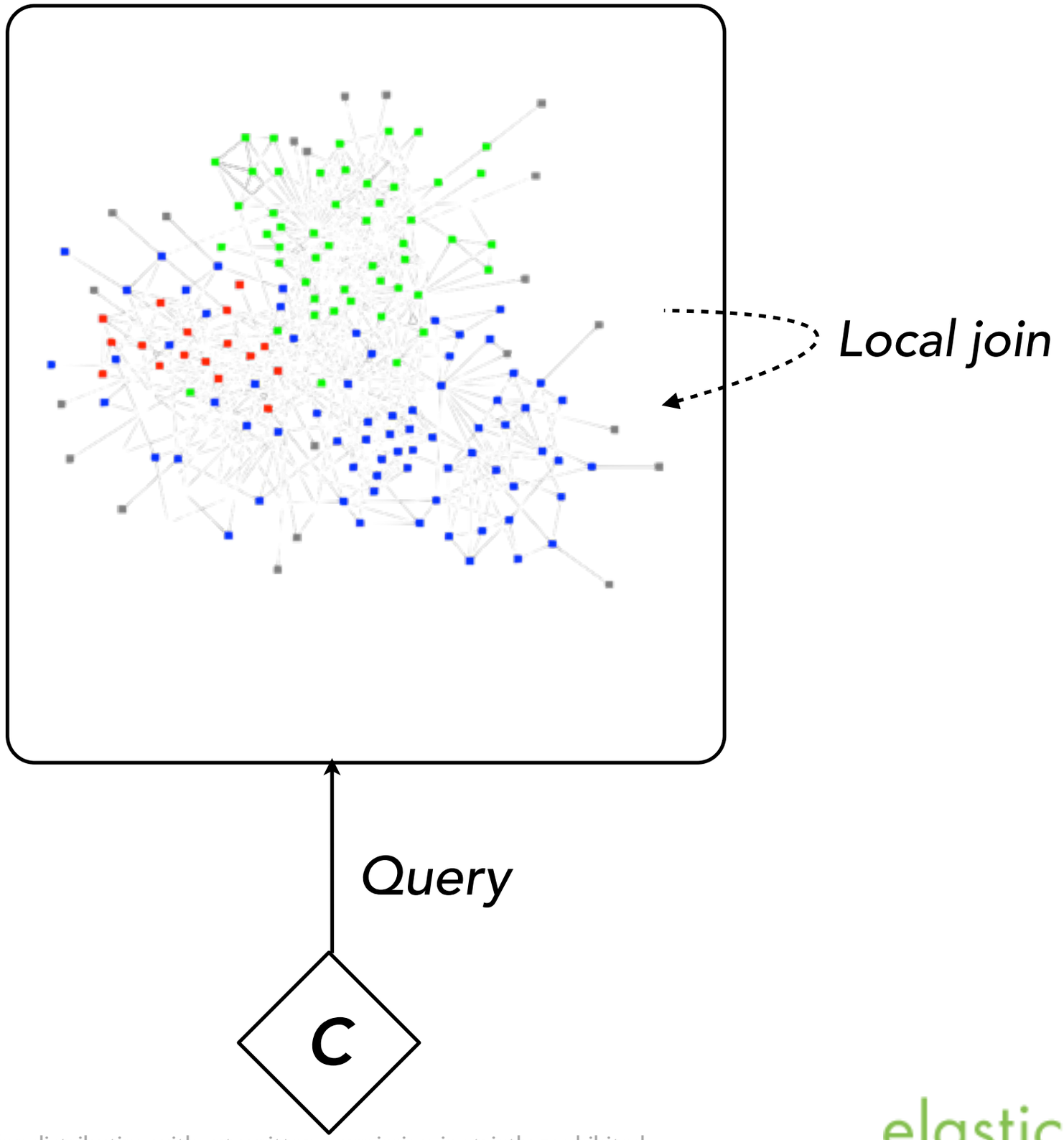
- Background
- Parent / child support
- Nested support
- Future developments

Background

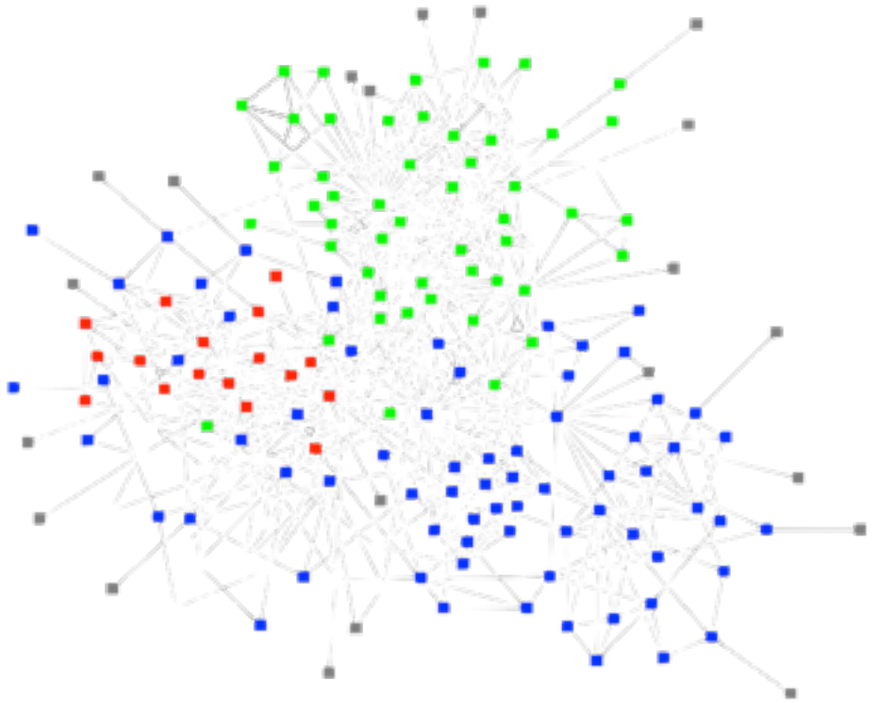


elasticsearch.

Background



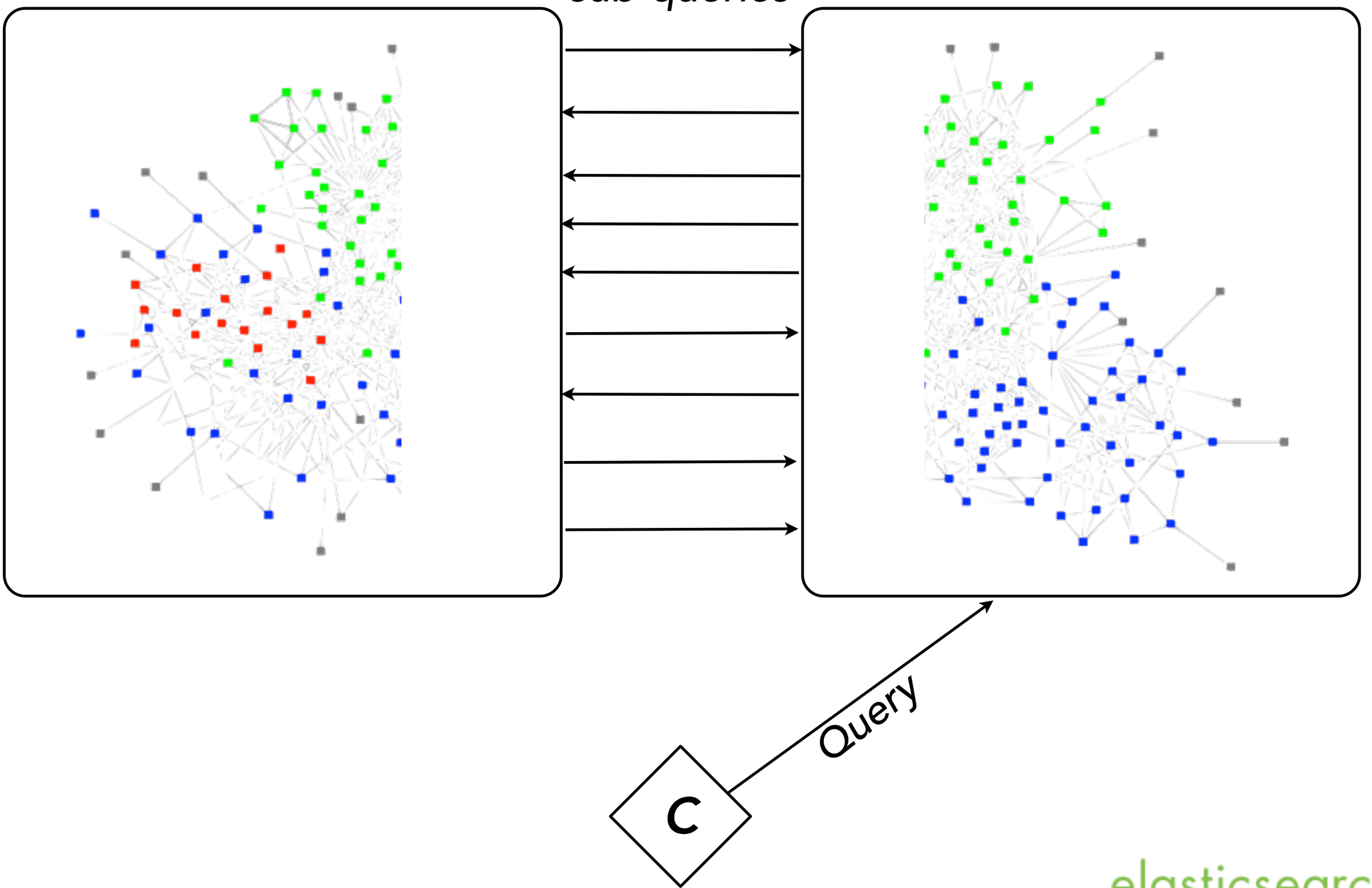
Background



- We need more capacity.
- But how to divide the relational data?

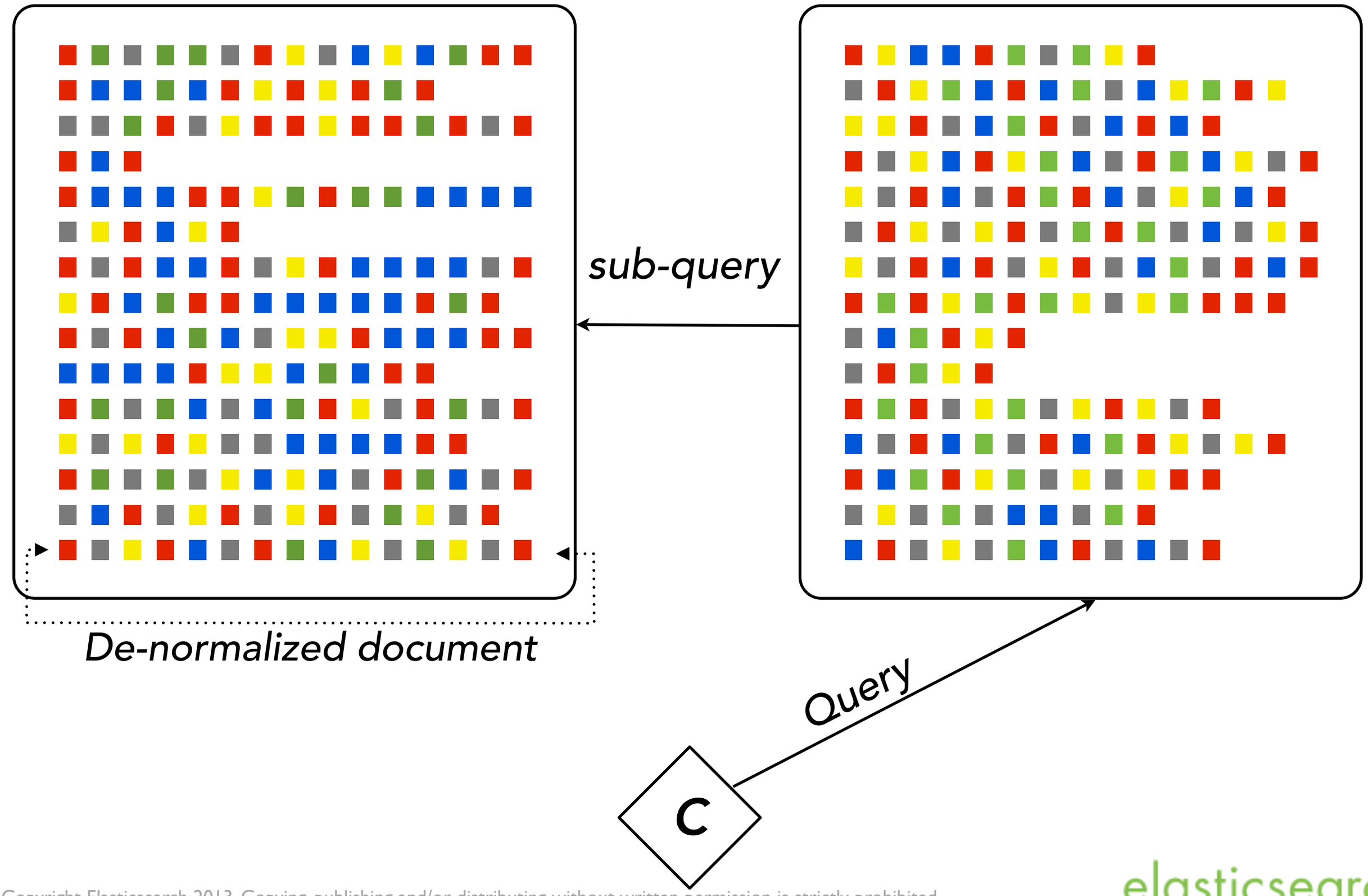
elasticsearch.

Background

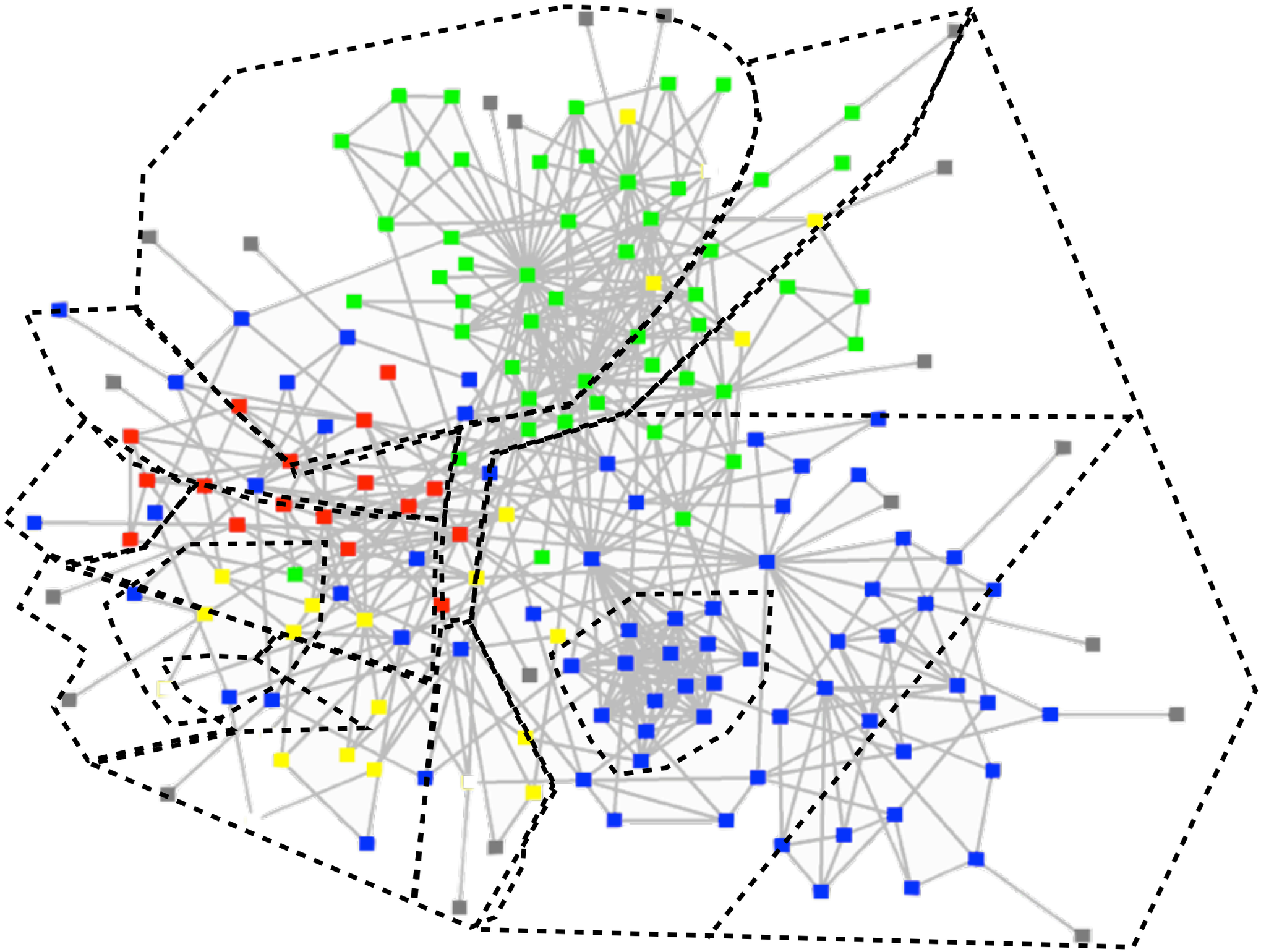


elasticsearch.

Background

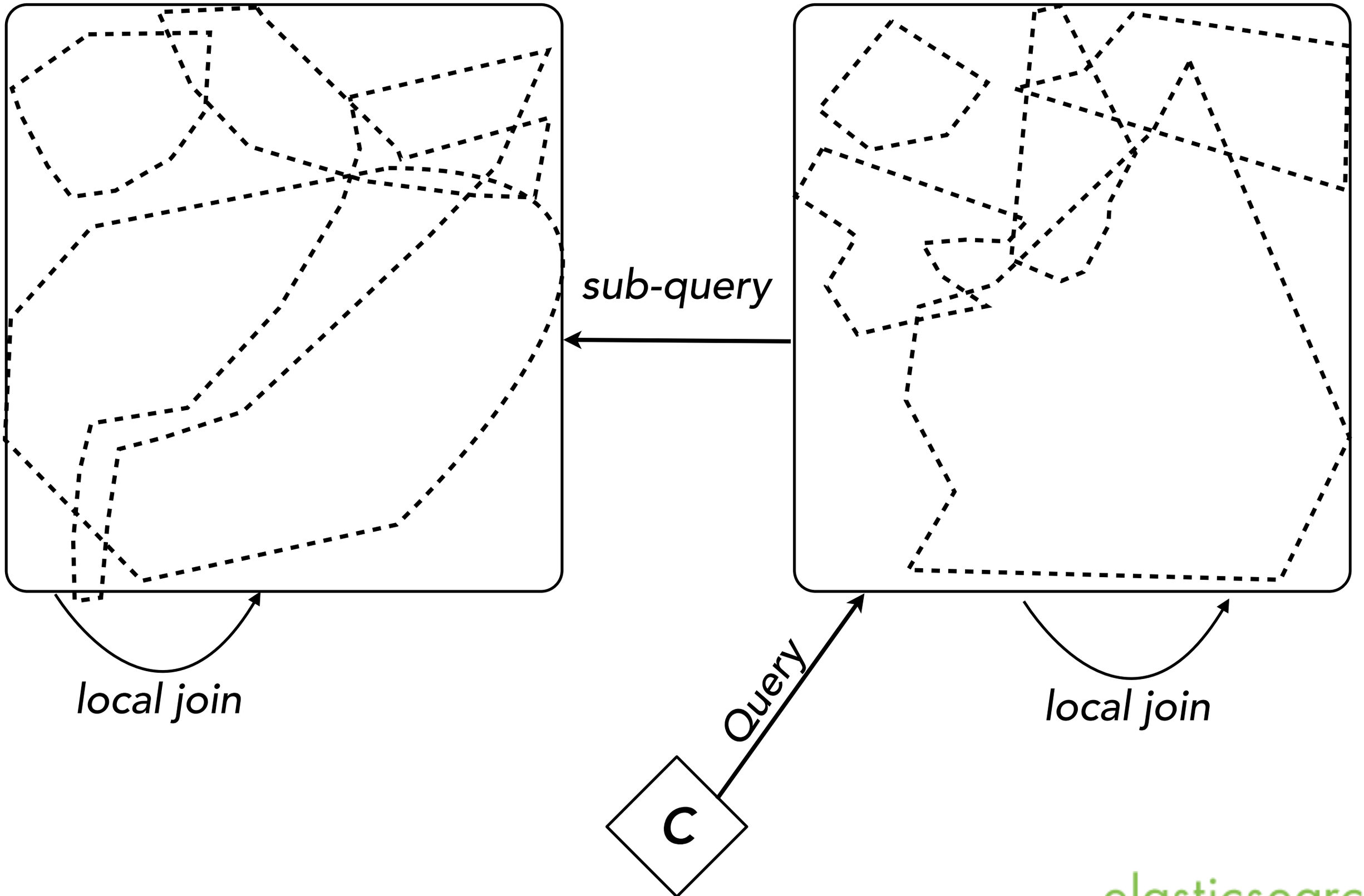


Background



elasticsearch.

Background



Background

- Dealing with relations either pay the price on write time or read time.

- Alternatively documents relations can balance the costs between read and write time.

  For example: one join to reduce duplicated data.

- Supporting “many-to-many” joins in a distributed system is difficult.

  Either unbalanced partitions or very expensive join.

Parent child

The query time join

elasticsearch.

Parent child

- Parent / child is a query time join between different document types in the same index.

- Parent and children documents are stored as separate documents in the same index.
  - Child documents can point to only one parent.
  - Parent documents can be referred by multiple child documents.

- Also a parent document can be a child document of a different parent.

elasticsearch.

Parent child

- A parent document and its children documents are routed into the same shard.
  - Parent id is used as routing value.

- In combination with a parent ids in memory data structure the parent-child join is fast.
  - Use warmer api to preload it!
  - Parent ids data structure size has significantly been reduced in version 0.90.1

Parent child - Indexing

A *offer* document is a parent of a product document

```
curl -XPUT 'localhost:9200/products' -d '{
   "mappings" : {
      "offer" : {
         "_parent" : { "type" : "product" }
      }
   }
}'
```

Then when indexing mention to what *product* a offer points to.

```
curl -XPUT 'localhost:9200/products/offer/12?parent=p2345' -d '{
  "valid_from" : "2013-05-01",
  "valid_to" : "2013-10-01",
  "price" : 26.87,
}'
```

- The parent document doesn't need to exist at time of indexing.

elasticsearch.

Parent child - Querying

- The ***has_child*** query returns parent documents based on matches in its child documents.

```
curl -XGET 'localhost:9200/products/_search' -d '{
    "query" : {
       "has_child" : {
          "type" : "offer",
          "query" : {
             "range" : {
                "price" : {
                   "lte" : 50
                }
             }
          }
       }
    }
}'
```

- The optional “score_mode” defines how child hits are mapped to its parent document.

Nested objects

The index time join

elasticsearch.

Nested objects

- In many cases domain models have the same write / update live-cycle.
  - Books & Chapters.
  - Movies & Actors.

- De-normalizing results in the fastest queries.
  - Compared to using parent/child queries.

- Nested objects allow smart de-normalization.

Nested objects

```
{
   "title" : "Elasticsearch",
   "authors" : "Clinton Gormley",
   "categories" : ["programming", "information retrieval"],
   "published_year" : 2013,
   "summary" : "The definitive guide for Elasticsearch ...",
   "chapter_1_title" : "Introduction",
   "chapter_1_summary" : "Short introduction about Elasticsearch's features ...",
   "chapter_1_number_of_pages" : 12,
   "chapter_2_title" : "Data in, Data out",
   "chapter_2_summary" : "How to manage your data with Elasticsearch ...",
   "chapter_2_number_of_pages" : 39,
   ...
}
```

Nested objects

```
{
   "title" : "Elasticsearch",
   "authors" : "Clinton Gormley",
   "categories" : ["programming", "information retrieval"],
   "published_year" : 2013,
   "summary" : "The definitive guide for Elasticsearch ...",
   "chapter_1_title" : "Introduction",
   "chapter_1_summary" : "Short introduction about Elasticsearch's features ...",
   "chapter_1_number_of_pages" : 12,
   "chapter_2_title" : "Data in, Data out",
   "chapter_2_summary" : "How to manage your data with Elasticsearch ...",
   "chapter_2_number_of_pages" : 39,
   ...
}
```

Too verbose!

elasticsearch.

Nested objects

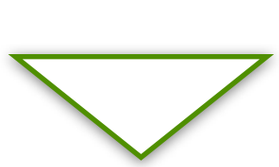
```
{
   "title" : "Elasticsearch",
   "author" : "Clinton Gormley",
   "categories" : ["programming", "information retrieval"],
   "published_year" : 2013,
   "summary" : "The definitive guide for Elasticsearch ...",
   "chapters" : [
      {
         "title" : "Introduction",
         "summary" : "Short introduction about Elasticsearch's features ...",
         "number_of_pages" : 12
      },
      {
         "title" : "Data in, Data out",
         "summary" : "How to manage your data with Elasticsearch ...",
         "number_of_pages" : 39
      },
      ...
   ]
}
```

- JSON allows complex nesting of objects.
- But how does this get indexed?

Nested objects

Original json document:

```
{
   "title" : "Elasticsearch",
   ...
   "chapters" : [
     {"title" : "Introduction", "summary" : "Short ...", "number_of_pages" : 12},
     {"title" : "Data in, ...", "summary" : "How to ...", "number_of_pages" : 39},
     ...
   ]
}
```



Lucene Document Structure:

```
{
   "title" : "Elasticsearch",
   ...
   "chapters.title" : ["Data in, Data out", "Introduction"],
   "chapters.summary" : ["How to ...", "Short ..."],
   "chapters.number_of_pages" : [12, 39]
}
```

Nested objects - Mapping

```
curl -XPUT 'localhost:9200/books' -d '{
  "mappings" : {
    "book" : {
      "properties" : {
        "chapters" : {
          "type" : "nested"
        }
      }
    }
  }
}'
```

Document type

Field type: ‘nested’

- The nested type triggers Lucene’s block indexing.

- Multiple levels of inner objects is possible.

elasticsearch.

Nested objects - Block indexing

Lucene Documents Structure:

```
{"chapters.title" : "Into...", "chapters.summary" : "...", "chapters.number_of_pages" : 12},
{"chapters.title" : "Data...", "chapters.summary" : "...", "chapters.number_of_pages" : 39},
...
{
  "title" : "Elasticsearch",
  ...
}
```

- Inlining the inner objects as separate Lucene documents right before the root document.
- The root document and its nested documents always remain in the same block.

Nested objects - Nested query

Specify the nested level.

score mode

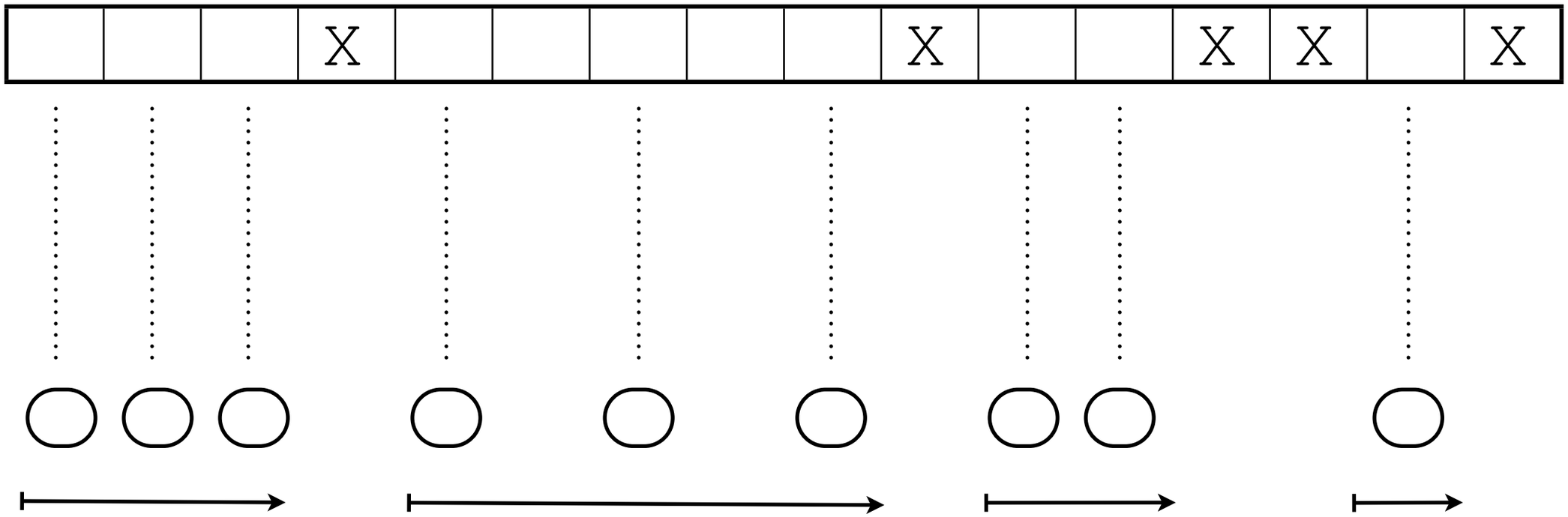
Chapter level query

```
curl -XGET 'localhost:9200/books/book/_search' -d '{
   "query" : {
      "nested" : {
         "path" : "chapters",
         "score_mode" : "avg",
         "query" : {
            "match" : {
               "chapters.summary" : {
                  "query" : "indexing data"
               }
            }
         }
      }
   }
}'
```

- Nested query returns the complete “book” as hit. (root document)

Nested objects

root documents bitset:



X Set bit, that represents a root document.

◯ Nested Lucene document, that match with the inner query.

⟼ Aggregate nested scores and push to root document.

elasticsearch.

Extra slides

But first questions!

elasticsearch.

Nested objects - Nested sorting

```
curl -XGET 'localhost:9200/books/book/_search' -d '{
  "query" : {
     "match" : {
         "summary" : {
            "query" : "guide"
         }
     }
  },
  "sort" : [
     {
        "chapters.number_of_pages" : {
           "sort_mode" : "avg",
           "nested_filter" : {
              "range" : {
                 "chapters.number_of_pages" : {"lte" : 15}
              }
           }
        }
     }
  ]
}'
```

Sort mode

elasticsearch.

Parent child - sorting

- Parent/child sorting isn't possible at the moment.
  - But there is a "custom_score" query work around.

```
"has_child" : {
   "type" : "offer",
   "query" : {
       "custom_score" : {
          "query" : { ... },
          "script" : "doc['price'].value"
       }
   }
}
```

- Downsides:
  - Forces to execute a script for each matching document.
  - The child sort value is converted into a float value.