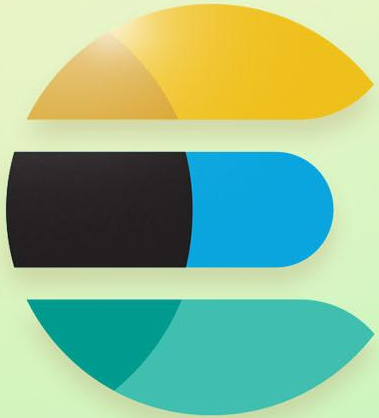
Querying by field existence



codingexplained.com

youtube.com/@codingexplained

CodingExplained

CODE. SLEEP. REPEAT.

Elasticsearch query

```
GET /products/_search
{
  "query": {
    "exists": {
      "field": "tags.keyword" (1)
    }
  }
}
```

(1) The `tags` mapping could just as well have been used

SQL query

```
SELECT *
FROM products
WHERE tags IS NOT NULL
```

How empty values are indexed

Field value	Indexed value
`NULL`	`N/A`
`[]`	`N/A`
`""`	`""`

Document #1

```
POST /products/_doc
{
  "name": "Protein Powder",
  "tags": ["Supplement"]
}
```

Document #2

```
POST /products/_doc
{
  "name": "Toast",
  "tags": []
}
```

Inverted index for `name` mapping

TERM	DOCUMENT #1	DOCUMENT #2
"protein"	X	
"powder"	X	
"toast"		X

Inverted index for `tags.keyword` mapping

TERM	DOCUMENT #1	DOCUMENT #2
"Supplement"	X	

Reasons for no indexed value

- Empty value provided (`NULL` or `[]`)
  - The `null_value` parameter is an exception for `NULL` values
- No value was provided for the field
- The `index` mapping parameter is set to `false` for the field
- The value's length is greater than the `ignore_above` parameter
- Malformed value with the `ignore_malformed` mapping parameter set to `true`

Inverting the query

```
GET /products/_search
{
  "query": {
    "bool": {
      "must_not": [
        {
          "exists": {
            "field": "tags.keyword"
          }
        }
      ]
    }
  }
}
```

SQL equivalent

```
SELECT * FROM products WHERE tags IS NULL
```

Lecture summary

- The `exists` query matches fields that have an *indexed* value
- Field values are only indexed if they are considered non-empty
  - `NULL` and empty arrays (`[]`) are empty values - empty strings (`""`) are not
  - There are a few other cases where values are not indexed
- The `exists` query can be inverted by using the `bool` query's `must_not` occurrence type