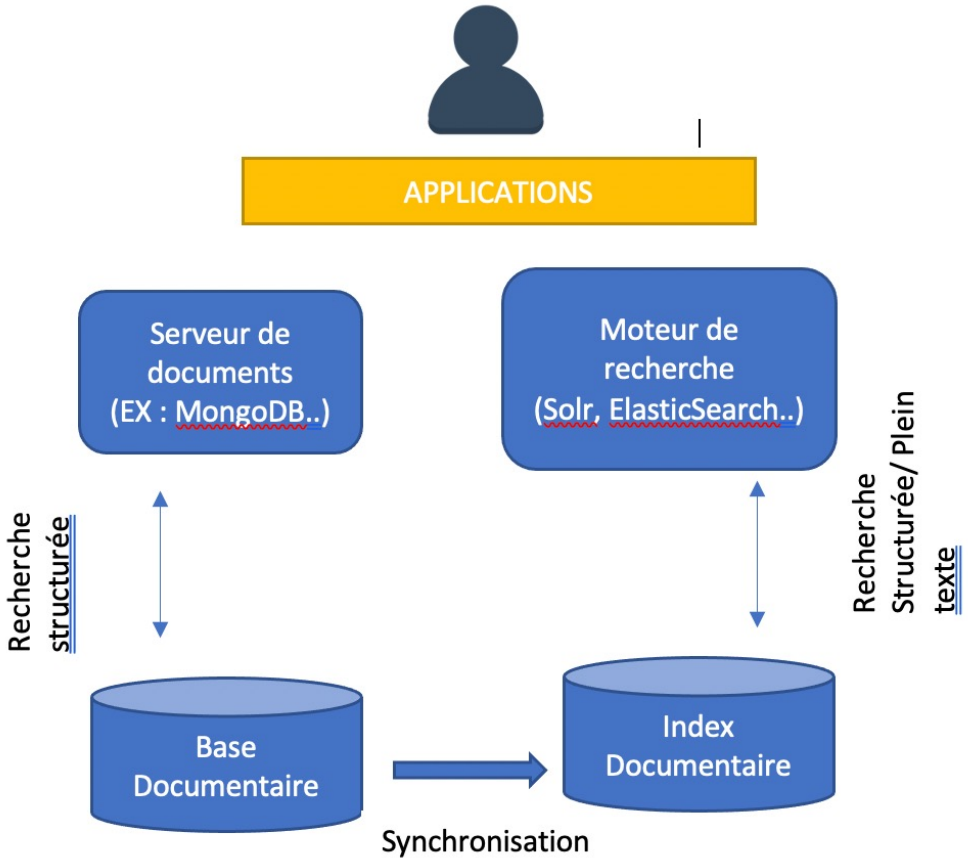
RECHERCHE D'INFORMATION

Master1 Miage Nanterre
Sonia GUEHIS

Plan

- Cours précédent:
  - *Définition de la RI*
  - *Stratégie et architecture*
  - *Evaluation et mesure de qualité: Rappel et Précision*
  - *Recherche plein texte: index inversé*
- Cours courant:
  - *Base documentaire et moteur de recherche*
  - *Pré-traitement des documents*
  - *ElsaticSearch: un exemple de moteur de recherche*

Bases documentaires et Moteur de recherche



PRÉ TRAITEMENT

- Effectuer un pré traitement sur les documents: une analyse des textes
- Effectuer une forme de normalisation pour se détacher de la forme du texte.
- Appliquer des transformations pour moins dépendre de la forme
- Exemple: un mot et ses variantes, la racine, le pluriel…
  - *Chanter, chant, chanteur, chanteurs….*

L'analyse de documents

- La normalisation est renforcée → La précision est fragilisée car:
  - Risque d'unifier des mots distincts.
  - Exemple : mère, mer

- La normalisation est renforcée → Le rappel est amélioré car:
  - Mise en correspondance des variantes d'un même terme avec un même sens
  - Exemple: joue, jouent, joues…

❖ Attention: Les transformations appliquées doivent être appliquées aux documents et aux requêtes , pour avoir une cohérence et éviter le décalage .

L'Analyse

L'analyse prend en considération quelques méta-données sur le document, tel que la langue, le contexte…

- Tokenization: découper le document en token « termes »
- Normalisation: les majuscules, les acronymes, les accents, les apostrophes…
  - *Exemple*: *Cours cours; l'arbre/les arbres; U.S.A/USA*

- Stemming (`racinisation`) , lemmatization:

Considérer la racine des mots pour éviter le biais de variations

  - *Exemple: manger, le manger, un mangeur…*
- Stop Words: ignorer des mots dans le document peu informatifs
  - *Exemple: la, et, de, à , des..*

Tokenisation

- Analyser le document et séparer le texte en tokens
  - *Exemple: Informatique, décisionnelle, requête, optimisation….*

- Problèmes rencontrés:
  - Que faire quand les mots dans le texte ne sont pas séparés par des espace (Ex: langue chinoise ou japonaise)
  - La gestion des langues s'écrivant de bas en haut, de droite à gauche.
  - Comment traiter les acronymes, les nombres, les unités, les URL…
  - Que faire avec les mots-composés : exemple: pomme de terre, aujourd'hui…en anglais: hostname, host-name, host name…
- Une technique de normalisation : minuscules et pas de ponctuation

Stemming ('racinisation') , lemmatization

- Confondre les mots d'une même forme, ou les mots apparentés en une seule racine
- Apparenter les marques de genre, les pluriels, les conjugaisons, les modes..
- Problèmes rencontrés:
  - C'est spécifique aux langues
  - Risque de confondre des mots d'un lexique proche: Exemple: *univers, université, universel..*
  - Fautes de frappes, d'orthographes..
- Exemple de technique de stemming: retirer les pluriels et mettre les verbes à l'infinitif.

Elimination des Stop Words

- Supprimer les mots peu informatifs pour limiter le stockage
  - Articles: le, la, ce, etc.
  - Verbes fonctionnels: être, avoir, faire..
  - Conjonctions: et, au, à …

- Cette technique n'est pas toujours utilisée car:
  - L'espace de stockage n'est plus une contrainte
  - Peut engendre des problèmes dans les mots composés: Exemple: pomme de terre, Arc de triomphe…

Exemple: document initial

d1: Le loup est dans la bergerie.

d2: Le loup et le trois petits cochons

d3: Les moutons sont dans la bergerie.

d4: Spider Cochon, Spider Cochon, il peut marcher au plafond.

d5: Un loup a mangé un mouton, les autres loups sont restés d ans la bergerie.

d6: Il y a trois moutons dans le pré, et un mouton dans la gueul e du loup.

d7: Le cochon est à 12 le Kg, le mouton à 10 Euro le kilo

d8: Les trois petits loups et le grand méchant cochon

1- Tokenisation:
mettre en minuscules et supprimer la ponctuation

d1 le loup est dans la bergerie

d2 le loup et le trois petits cochons

d3 les moutons sont dans la bergerie

d4 spider cochon spider cochon il peut marcher au plafond

d5 un loup a mangé un mouton les autres loups sont restés da ns la bergerie

d6 il y a trois moutons dans le pré et un mouton dans la gueule du loup

d7 le cochon est à 12 le Kg le mouton à 10 euro le kilo

d8 les trois petits loups et le grand méchant cochon

2- Stemming:

Retirer les pluriels et mettre à l'infinitif

d1 le loup etre dans la bergerie

d2 le loup et les trois petit cochon

d3 les mouton etre dans la bergerie

d4 spider cochon spider cochon il pouvoir marcher au plafond

d5 un loup avoir
manger un mouton les autre loup etre rester dans la bergerie

d6 il y avoir trois mouton dans le pré et un mouton dans la gueule du loup

d7 le cochon etre a 12 le Kg le mouton à 10 euro le kilo

d8 les trois petit loup et le grand mechant cochon

3- Stop Words:

Retirer les mots peu informatifs

d1 loup etre bergerie
d2 loup trois petit cochon
d3 mouton etre bergerie
d4 spider cochon spider cochon pouvoir marcher plafond
d5 loup avoir manger mouton autre loup etre rester bergerie
d6 avoir trois mouton pré mouton gueule loup
d7 cochon etre 12 Kg mouton 10 euro kilo
d8 trois petit loup grand mechant cochon

ELASTICSEARCH

elasticsearch

ElasticSearch

ElasticSearch est :

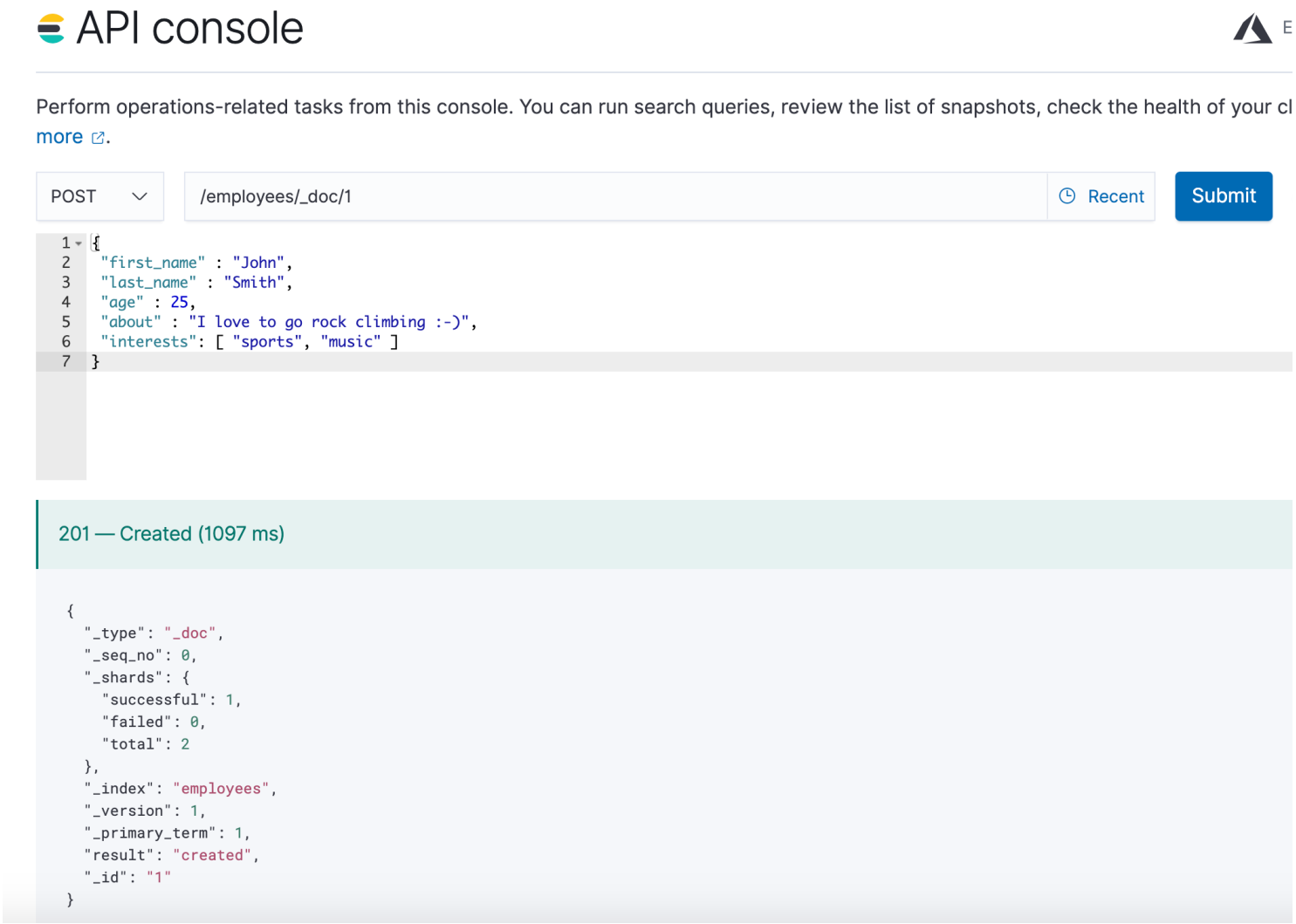
- un serveur de recherche avancé
- open source
- basé sur Lucene
- écrit en Java.
- fournit des fonctionnalités de recherche distribuées, en texte intégral ou partiel, basées sur la requête et la géolocalisation
- accessible via une API HTTP REST, généralement via la bibliothèque cURL.
- les messages entre le serveur de recherche et le client sont envoyés sous la forme de chaînes JSON
- Par défaut, ElasticSearch s'exécute sur le port 9200.

ElasticSearch : Installation

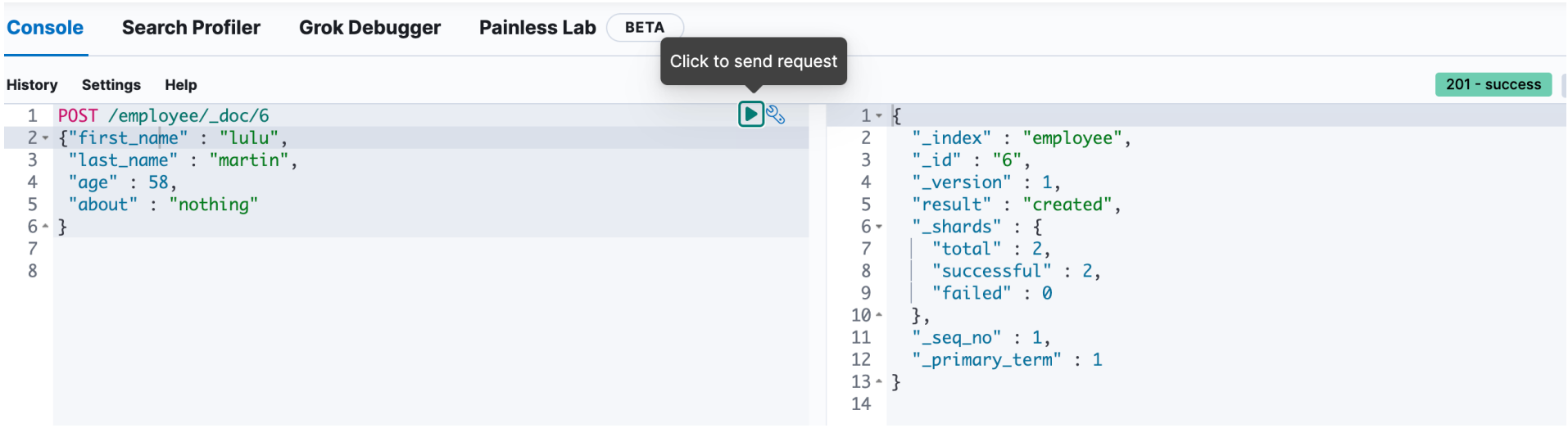
- ElasticSearch peut être installé sous plusieurs OS possibles
- Les primitives d'installation pour Linux et Windows sont décrites dans le livre :
  - ✓ https://riptutorial.com/Download/elasticsearch-fr.pdf
- Nous optons dans le cadre de ce cours d'utiliser la version démo d'ElasticSearch on the cloud, accessible en ligne :
  - ✓ https://cloud.elastic.co/home

  Vous devez renseigner une adresse email et un mot de passe.
- Vous avez alors accès à l'API console, qui va nous permettre d'exécuter les requêtes elastic.

ElasticSearch : Ajout d'un employé



ElasticSearch : Ajout d'un employé



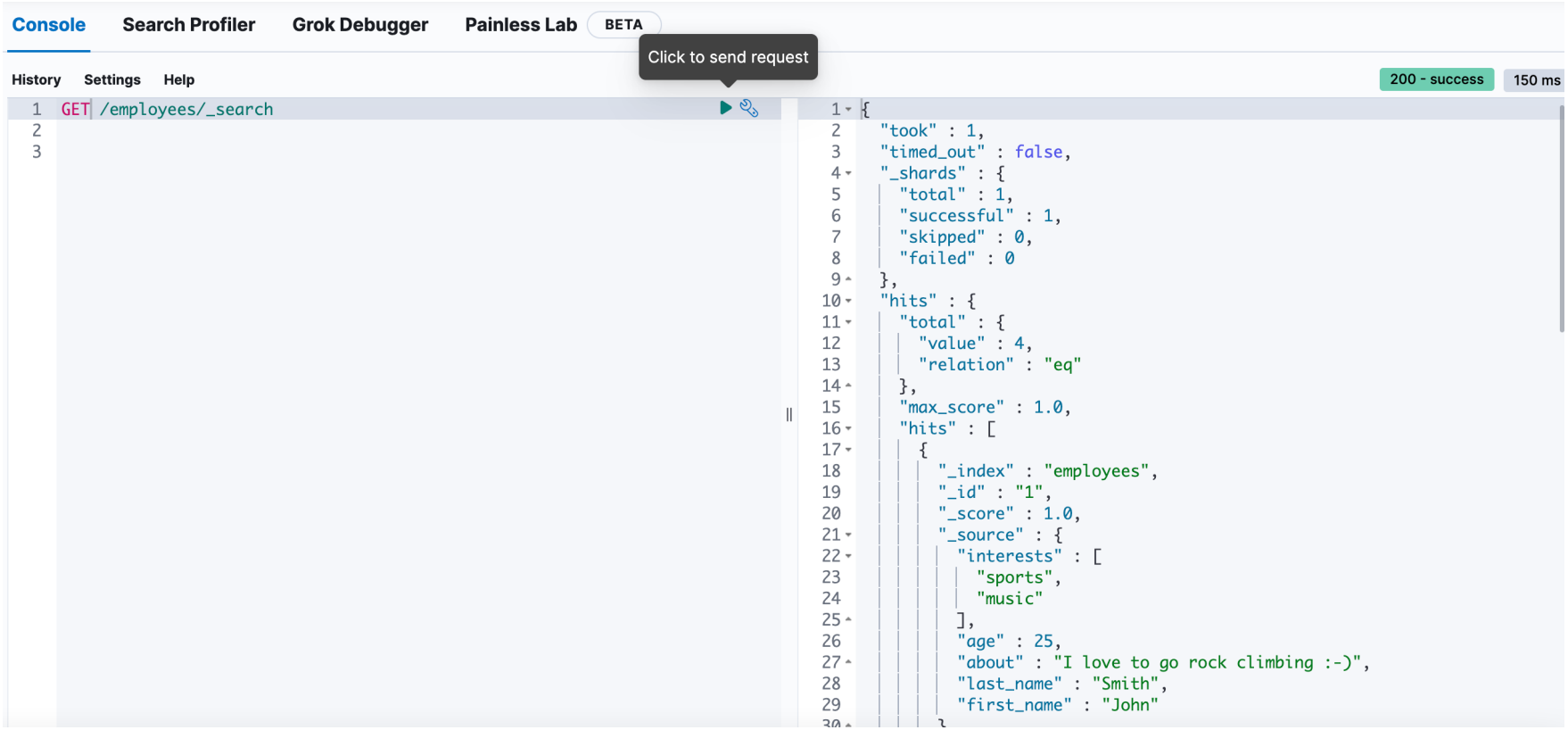
ElasticSearch : Consulter les documents

GET /employees/_search

200 — OK (463 ms)


```
{
  "hits": {
    "hits": [
      {
        "_score": 1,
        "_type": "_doc",
        "_id": "1",
        "_source": {
          "interests": [
            "sports",
            "music"
          ],
          "age": 25,
          "last_name": "Smith",
          "about": "I love to go rock climbing :-)",
          "first_name": "John"
        },
        "_index": "employees"
      },
      {
        "_score": 1,
        "_type": "_doc",
        "_id": "2",
        "_source": {
          "interests": [
            "music"
          ],
          "age": 32,
          "last_name": "Smith",
          "about": "I like to collect rock albums :-)",
          "first_name": "Jane"
        },
        "_index": "employees"
      }
    ],
    "total": {
      "relation": "eq",
      "value": 2
    },
    "max_score": 1
  },
  "_shards": {
    "successful": 1,
    "failed": 0,
    "skipped": 0,
    "total": 1
  },
```

ElasticSearch : Consulter les documents



GET /employees/

200 — OK (536 ms)


```
{
  "employees": {
    "settings": {
      "index": {
        "provided_name": "employees",
        "number_of_replicas": "1",
        "uuid": "gvB5iivuTJChzP9HZ6le_w",
        "number_of_shards": "1",
        "creation_date": "1615162039433",
        "version": {
          "created": "7110199"
        },
        "routing": {
          "allocation": {
            "include": {
              "_tier_preference": "data_content"
            }
          }
        }
      }
    },
    "mappings": {
      "properties": {
        "interests": {
          "fields": {
            "keyword": {
              "ignore_above": 256,
              "type": "keyword"
            }
          },
          "type": "text"
        },
        "about": {
          "fields": {
            "keyword": {
              "ignore_above": 256,
              "type": "keyword"
            }
          },
          "type": "text"
        },
        "last_name": {
          "fields": {
            "keyword": {
              "ignore_above": 256,
              "type": "keyword"
            }
          },
          "type": "text"
        },
        "age": {
          "type": "long"
        },
        "first_name": {
          "fields": {
            "keyword": {
              "ignore_above": 256,
              "type": "keyword"
            }
          },
          "type": "text"
        }
      }
    },
    "aliases": {}
  }
}
```

ElasticSearch : Suppression

DELETE /employees/_doc/1

```
1
```

200 — OK (241 ms)


```
{
  "_type": "_doc",
  "_seq_no": 2,
  "_shards": {
    "successful": 2,
    "failed": 0,
    "total": 2
  },
  "_index": "employees",
  "_version": 2,
  "_primary_term": 1,
  "result": "deleted",
  "_id": "1"
}
```

Le mapping

- ElasticSearch possède un mapping.
- Le mapping détermine comment les algorithmes doivent interpréter une entrée.
- Tous les champs sont affectés aux types text et keyword.
- Elasticsearch reconnaît toutefois 6 types de données de base et encore plus de domaines spécifiques. Les 6 principaux types sont divisés en partie en d'autres sous-catégories :

Les types principaux

- **String** : text et keyword. Bien que Keywords est pris en compte comme correspondance exacte, Elasticsearch part du principe qu'un texte doit être analys avant de pouvoir être utilisé.
- **Numeric** : Elasticsearch reconnaît des valeurs numériques qui diffèrent surtout par leur étendue. Par exemple, alors que le type byte peut avoir des valeurs comprises entre -128 et 127, on considérera avec long une fourchette allant de -2^{63} à $2^{63}-1$.
- **Date** : une date peut être spécifiée au jour ou à l'heure.
- **Boolean** : les champs formatés boolean peuvent avoir une valeur vraie (true) ou fausse (false).
- **Binary** : vous pouvez insérer des données binaires dans ces champs. Pour cela, utilisez l'encodage Base64.
- **Range** : il permet de spécifier une plage. Elle peut se situer entre deux valeurs numériques, deux données ou même entre deux adresses IP.

ElasticSearch : Consulter les mappings

GET /employees/_mapping


```
{
  "employees": {
    "mappings": {
      "properties": {
        "interests": {
          "fields": {
            "keyword": {
              "ignore_above": 256,
              "type": "keyword"
            }
          },
          "type": "text"
        },
        "about": {
          "fields": {
            "keyword": {
              "ignore_above": 256,
              "type": "keyword"
            }
          },
          "type": "text"
        },
        "last_name": {
          "fields": {
            "keyword": {
              "ignore_above": 256,
              "type": "keyword"
            }
          },
          "type": "text"
        },
        "age": {
          "type": "long"
        },
        "first_name": {
          "fields": {
            "keyword": {
              "ignore_above": 256,
              "type": "keyword"
            }
          },
          "type": "text"
        }
      }
    }
  }
}
```

ElasticSearch : Rechercher un match exact

POST /employees/_search

```
{
 "query" : {
 "match" : {
 "age" : 32
 }
 }
}
```


```
{
  "hits": {
    "hits": [
      {
        "_score": 1,
        "_type": "_doc",
        "_id": "2",
        "_source": {
          "interests": [
            "music"
          ],
          "age": 32,
          "last_name": "Smith",
          "about": "I like to collect rock albums :-)",
          "first_name": "Jane"
        },
        "_index": "employees"
      }
    ],
    "total": {
      "relation": "eq",
      "value": 1
    },
    "max_score": 1
  },
  "_shards": {
    "successful": 1,
    "failed": 0,
    "skipped": 0,
    "total": 1
  },
  "took": 3,
  "timed_out": false
}
```

ElasticSearch :

- « Par défaut, le document indexé complet est renvoyé dans le cadre de toutes les recherches. C'est ce que l'on appelle la source (champ _source dans les résultats de recherche).

- Si nous ne voulons pas que tout le document source soit retourné, nous avons la possibilité de ne demander que quelques champs de la source à retourner, ou nous pouvons définir _source sur false pour omettre complètement le champ. » [1]

Récupérer des propriétés uniquement:
« Age et about »

POST /employees/_search

```
{
 "query": { "match_all": {} },
 "_source": ["age", "about"]
}
```


```
{
  "hits": {
    "hits": [
      {
        "_score": 1,
        "_type": "_doc",
        "_id": "1",
        "_source": {
          "about": "I love to go rock climbing :-)",
          "age": 25
        },
        "_index": "employees"
      },
      {
        "_score": 1,
        "_type": "_doc",
        "_id": "2",
        "_source": {
          "about": "I like to collect rock albums :-)",
          "age": 32
        },
        "_index": "employees"
      }
    ],
    "total": {
      "relation": "eq",
      "value": 2
    },
    "max_score": 1
  },
  "_shards": {
    "successful": 1,
    "failed": 0,
    "skipped": 0,
    "total": 1
  },
  "took": 3,
  "timed_out": false
}
```

Ajouter un filtre
Avoir ‘music’ ou ‘sports’ dans interests

POST /employees/_search

```
{
 "query": { "match": { "interests": "music sports" } }
}
```


```
{
  "hits": {
    "hits": [
      {
        "_score": 0.77041256,
        "_type": "_doc",
        "_id": "1",
        "_source": {
          "interests": [
            "sports",
            "music"
          ],
          "age": 25,
          "last_name": "Smith",
          "about": "I love to go rock climbing :-)",
          "first_name": "John"
        },
        "_index": "employees"
      },
      {
        "_score": 0.21110919,
        "_type": "_doc",
        "_id": "2",
        "_source": {
          "interests": [
            "music"
          ],
          "age": 32,
          "last_name": "Smith",
          "about": "I like to collect rock albums :-)",
          "first_name": "Jane"
        },
        "_index": "employees"
      }
    ],
    "total": {
      "relation": "eq",
      "value": 2
    },
    "max_score": 0.77041256
  },
  "_shards": {
    "successful": 1,
    "failed": 0,
    "skipped": 0,
    "total": 1
  },
  "took": 3,
  "timed_out": false
}
```

Ajouter un filtre avec « bool/should »
Avoir ‘rocks’ ou ‘albums’ dans about

POST /employees/_search

```
{
 "query": {
 "bool": {
 "should": [
  { "match": { "about": "rock" } },
  { "match": { "about": "albums" } }
  ]
  }
  }
}
```


```
{
  "hits": {
    "hits": [
      {
        "_score": 0.8754687,
        "_type": "_doc",
        "_id": "2",
        "_source": {
          "interests": [
            "music"
          ],
          "age": 32,
          "last_name": "Smith",
          "about": "I like to collect rock albums :-)",
          "first_name": "Jane"
        },
        "_index": "employees"
      },
      {
        "_score": 0.18232156,
        "_type": "_doc",
        "_id": "1",
        "_source": {
          "interests": [
            "sports",
            "music"
          ],
          "age": 25,
          "last_name": "Smith",
          "about": "I love to go rock climbing :-)",
          "first_name": "John"
        },
        "_index": "employees"
      }
    ],
    "total": {
      "relation": "eq",
      "value": 2
    },
    "max_score": 0.8754687
  },
  "_shards": {
    "successful": 1,
    "failed": 0,
    "skipped": 0,
    "total": 1
  },
  "took": 4,
  "timed_out": false
}
```

Ajouter un filtre avec « bool /must»
Avoir ‘rocks’ et ‘albums’ dans about

POST /employees/_search

```
{
 "query": {
 "bool": {
 "must": [
  { "match": { "about": "rock" } },
  { "match": { "about": "albums" } }
  ]
  }
  }
}
```


```
{
  "hits": {
    "hits": [
      {
        "_score": 0.8754687,
        "_type": "_doc",
        "_id": "2",
        "_source": {
          "interests": [
            "music"
          ],
          "age": 32,
          "last_name": "Smith",
          "about": "I like to collect rock albums :-)",
          "first_name": "Jane"
        },
        "_index": "employees"
      }
    ],
    "total": {
      "relation": "eq",
      "value": 1
    },
    "max_score": 0.8754687
  },
  "_shards": {
    "successful": 1,
    "failed": 0,
    "skipped": 0,
    "total": 1
  },
  "took": 3,
  "timed_out": false
}
```

Ajouter un filtre avec « bool /must / must not»

Avoir ‘music’ et ne pas avoir 32 ans

POST /employees/_search

```
{
 "query": {
 "bool": {
 "must": [
 { "match": { "interests": "music" } }
 ],
 "must_not": [
 { "match": { "age": "32" } }
 ]
 }
 }
}
```


```
{
  "hits": {
    "hits": [
      {
        "_score": 0.160443,
        "_type": "_doc",
        "_id": "1",
        "_source": {
          "interests": [
            "sports",
            "music"
          ],
          "age": 25,
          "last_name": "Smith",
          "about": "I love to go rock climbing :-)",
          "first_name": "John"
        },
        "_index": "employees"
      }
    ],
    "total": {
      "relation": "eq",
      "value": 1
    },
    "max_score": 0.160443
  },
  "_shards": {
    "successful": 1,
    "failed": 0,
    "skipped": 0,
    "total": 1
  },
  "took": 1,
  "timed_out": false
}
```

Ajouter un filtre: Avoir une chaine exacte
Match_phrase

POST /employees/_search

```
{
 "query": {
     "match_phrase": { "about": "collect rock albums " } }
}
```


```
{
  "hits": {
    "hits": [
      {
        "_score": 1.5686159,
        "_type": "_doc",
        "_id": "2",
        "_source": {
          "interests": [
            "music"
          ],
          "age": 32,
          "last_name": "Smith",
          "about": "I like to collect rock albums :-)",
          "first_name": "Jane"
        },
        "_index": "employees"
      }
    ],
    "total": {
      "relation": "eq",
      "value": 1
    },
    "max_score": 1.5686159
  },
  "_shards": {
    "successful": 1,
    "failed": 0,
    "skipped": 0,
    "total": 1
  },
  "took": 1,
  "timed_out": false
}
```

Aggrégation: avg

POST /employees/_search

```
{
 "aggs" : {
 "avd_value" : { "avg" : { "field" : "age" } }
 },
  "_source": ["age"]
}
```


```
{
  "hits": {
    "hits": [
      {
        "_score": 1,
        "_type": "_doc",
        "_id": "1",
        "_source": {
          "age": 25
        },
        "_index": "employees"
      },
      {
        "_score": 1,
        "_type": "_doc",
        "_id": "2",
        "_source": {
          "age": 32
        },
        "_index": "employees"
      }
    ],
    "total": {
      "relation": "eq",
      "value": 2
    },
    "max_score": 1
  },
  "_shards": {
    "successful": 1,
    "failed": 0,
    "skipped": 0,
    "total": 1
  },
  "took": 1,
  "aggregations": {
    "avd_value": {
      "value": 28.5
    }
  },
  "timed_out": false
}
```

ELASTICSEARCH : Analyseur

Les Analyseurs

- L'analyse: le processus qu'Elasticsearch exécute sur le corps d'un document avant que celui-ci ne soit envoyé pour être ajouté à l'index inversé.

- Elasticsearch passe par plusieurs étapes pour chaque champ à analyser avant l'ajout de l'index au document :
  - Filtre de caractères : transforme les caractères à l'aide d'un filtre de caractères
  - Découper le texte en jetons: découpe le texte en un ou plusieurs jetons.
  - Filtre de jetons: transforme chaque jeton à l'aide d'un filtre de jetons
  - Indexation de jetons: stocke ces jetons dans l'index

Les Analyseurs

- Il existe deux manières de spécifier des analyseurs pouvant être utilisés par vos champs:
  - Lorsque l'index est créé, en tant que paramètres pour cet index particulier
  - En tant qu'analyseurs globaux dans le fichier de configuration pour Elasticsearch

- Les analyseurs permettent de 'tokeniser' le texte d'un champ de chaîne de caractères. Ces tokens seront utiles lors des recherches.

- L'analyseur peut être appliqué aux 'mappings' de sorte que, lorsque les champs sont indexés, ils soient effectués sur chaque terme plutôt que sur la chaîne globale.

- Au moment des recherches, la chaîne d'entrée sera également soumise à l'analyseur. Par conséquent, si vous normalisez une entrée dans l'analyseur, elle matchera même si la requête contient une chaîne non normalisée.[1]

Les Analyseurs

- Un analyseur peut être appliqué à un mapping en utilisant "analyse »
- L'analyseur "standard" est utilisé par défaut. Si vous ne souhaitez pas utiliser d'analyseur (parce que la segmentation ou la normalisation ne serait pas utile), vous pouvez spécifier "index": "not_analyzed »
- Le tokenizer standard: un des nombreux tokenizer fournis par Elasticsearch
- Le tokenizer standard: adapté à la plupart des langues basées sur l'alphabet latin et est conçu pour être assez polyvalent pour traiter différents types de texte.
- Le tokenizer standard: souvent utilisé dans des cas où le texte à analyser contient une langue naturelle (comme l'anglais, le français, etc.) et où il n'y a pas de nécessité de conserver des symboles ou des numéros spéciaux comme tokens séparés. C'est un choix judicieux pour les analyses de base du texte car il fournit un bon équilibre entre efficacité et précision.

Les Analyseurs

- Un analyseur fonctionne dans une séquence:

  - Tokenizer (one):

https://www.elastic.co/guide/en/elasticsearch/reference/current/analysis-tokenizers.html

  - TokenFilters (zéro ou plus) :

Après la tokenisation, on peut appliquer des filtres pour modifier les tokens. Exemple lowercasing, remove stopwords, ajouter des tokens (les synonymes)…
https://www.elastic.co/guide/en/elasticsearch/reference/current/analysis-tokenfilters.html

  - WhitespaceTokenizer:

Consiste à découper le texte en tokens en se basant sur les espaces blancs uniquement.

https://www.elastic.co/guide/en/elasticsearch/reference/current/analysis-whitespace-tokenizer.html

Exemple:

```
PUT my_index {
        "mappings": {
                "user": {
                        "properties": {
                                "name": { "type": "string »
                                "analyzer": "my_user_name_analyzer »
                                        },
                        "id": {
                                "type": "string",
                                "index": "not_analyzed »
                                }
                         }
                }
        }
}
```

Exemples et tests des anlyseurs

POST /employees/_analyze

```
{
  "tokenizer": "standard",
  "filter":  [ "lowercase", "asciifolding" ],
  "text":      "Is this déja VU?"
}
```

```
{
  "tokens": [
    {
      "end_offset": 2,
      "token": "is",
      "type": "<ALPHANUM>",
      "start_offset": 0,
      "position": 0
    },
    {
      "end_offset": 7,
      "token": "this",
      "type": "<ALPHANUM>",
      "start_offset": 3,
      "position": 1
    },
    {
      "end_offset": 12,
      "token": "deja",
      "type": "<ALPHANUM>",
      "start_offset": 8,
      "position": 2
    },
    {
      "end_offset": 15,
      "token": "vu",
      "type": "<ALPHANUM>",
      "start_offset": 13,
      "position": 3
    }
  ]
}
```

Tokenizer: « keyword » ne pas diviser en token

POST /employees/_analyze

```
1
2 {
3   "tokenizer": "keyword",
4   "filter": [ "lowercase" ],
5   "text": "john.SMITH@example.COM"
6 }
```

200 — OK (502 ms)

```
{
  "tokens": [
    {
      "end_offset": 22,
      "token": "john.smith@example.com",
      "type": "word",
      "start_offset": 0,
      "position": 0
    }
  ]
}
```

Créer un index avec un pattern:
Configurer le simple_pattern tokenizer à produire les termes a 3

PUT my-index-000001

```
{
  "settings": {
    "analysis": {
      "analyzer": {
        "my_analyzer": {
          "tokenizer": "my_tokenizer"
        }
      },
      "tokenizer": {
        "my_tokenizer": {
          "type": "simple_pattern",
          "pattern": "[0123456789]{3}"
        }
      }
    }
  }
}
```

POST my-index-000001/_analyze

```
{
  "analyzer": "my_analyzer",
  "text": "fd-786-35-514-x"
}
```

200 — OK (172 ms)

```
{
  "tokens": [
    {
      "end_offset": 6,
      "token": "786",
      "type": "word",
      "start_offset": 3,
      "position": 0
    },
    {
      "end_offset": 13,
      "token": "514",
      "type": "word",
      "start_offset": 10,
      "position": 1
    }
  ]
}
```

Stemmer:

POST _analyze

```
{
  "tokenizer": "standard",
  "filter": [ "stemmer" ],
  "text": "the foxes jumping quickly"
}
```


```
{
  "tokens": [
    {
      "end_offset": 3,
      "token": "the",
      "type": "<ALPHANUM>",
      "start_offset": 0,
      "position": 0
    },
    {
      "end_offset": 9,
      "token": "fox",
      "type": "<ALPHANUM>",
      "start_offset": 4,
      "position": 1
    },
    {
      "end_offset": 17,
      "token": "jump",
      "type": "<ALPHANUM>",
      "start_offset": 10,
      "position": 2
    },
    {
      "end_offset": 25,
      "token": "quickli",
      "type": "<ALPHANUM>",
      "start_offset": 18,
      "position": 3
    }
  ]
}
```

Références Bibliographiques

1. Apprenez ElasticSearch, ebook gratuit non affilié crée à partir des contributeurs de Stack Overflow

2. https://www.elastic.co/guide/en/elasticsearch/reference/current/analysis.html

3. http://b3d.bdpedia.fr/ri-tpelasticdsl.html