

# TP Elasticserach

@SONIA GUEHIS



# ElasticSearch is

- ✓ « a **real-time distributed** search and analytics engine with **high availability**.
- ✓ Used for full-text search, structured search, analytics or all three in combination
- ✓ Built on top of the Apache Lucence library
- ✓ It's **schema-free, document-oriented** data store » [1]

# Terminology [1]

SQL	Elasticsearch	Description
Column	<b>Field</b>	A field is the smallest unit of data in ElasticSearch. It can contain a list of multiple values of the same type
Row	<b>Document</b>	A document is like a row to group fields (columns in SQL). Is a JSON object in Elasticsearch
Table	<b>Index</b>	An index is the largest unit of data in Elasticsearch. An index is a logical partition of the indexed documents and the target against which the search queries get executed.
Schema	<b>Implicit</b>	Elasticsearch does not provide an equivalent concept for it.

# Le mapping

- ▶ Définit comment un document et les champs qu'il contient, sont stockés et indexés.
- ▶ Peut être défini explicitement pour un index lors de sa création. Cela permet de spécifier les types de champs (par exemple, text, date, integer, etc.), si un champ doit être indexé, et d'autres configurations spécifiques au champ.
- ▶ Le mapping d'un index existant peut être mis à jour pour ajouter de nouveaux champs. Cependant, il y a des limitations à ce que vous pouvez modifier pour les champs existants (par exemple, vous ne pouvez pas changer le type de données d'un champ).
- ▶ Elasticsearch peut automatiquement déterminer le mapping basé sur les données des documents lorsqu'ils sont indexés. Cela peut être utile pour des cas d'utilisation simples, mais pour un contrôle plus précis et une optimisation, il est recommandé de définir explicitement le mapping.

```
PUT /mon_index
```

```
{
  "mappings": {
    "properties": {
      "nom": {
        "type": "text"
      },
      "age": {
        "type": "integer"
      },
      "email": {
        "type": "keyword"
      },
      "date_inscription": {
        "type": "date"
      }
    }
  }
}
```

- Le type **keyword** dans Elasticsearch est utilisé pour stocker des données de texte exactes, non analysées ou divisées en tokens lors de l'indexation. Ex: les identifiants, les adresses email, les tags, les étiquettes, et d'autres textes qui doivent être utilisés pour des recherches exactes, des filtres, des tris, ou des agrégations



# L'API de recherche

- Recherche en utilisant le corps de la requête

GET /my\_index/type/\_search

```
{  
  "query" : {  
    "term" : { "field_to_search" : "search_item" }  
  }  
}
```

# L'API de recherche

- La multi recherche: rechercher une requête dans plusieurs champs à la fois:

```
GET /_search {  
  "query": {  
    "multi_match" : {  
      "query": "text to search",  
      "fields": [ "field_1", "field_2" ]  
    }  
  }  
}
```

- Lister tous les indexes:

```
GET /_cat/indices?v
```

- Créer un index avec un nom « car »

```
PUT /car?pretty
```

- Pour indexer le document avec nom « car » en utilisant l'id 1

```
PUT /car/_doc/1?pretty=true
```

```
{  
  "field": "value"  
}
```

- Pour récupérer ce document:

```
GET /car/_doc/1?pretty
```

PS: Lorsque vous ajoutez ?pretty=true à la fin de votre URL de requête, Elasticsearch formate le JSON de réponse avec des indentations et des retours à la ligne.



- Pour mettre à jour le document:

```
PUT /car/_doc/1?pretty
{
  "name": "Tata Nexa"
}
```

- Indexer un document sans identifiant (identifiant implicite)

```
POST /car/_doc?pretty
{
  "name": "Jane Doe"
}
```

- Afficher tous les documents de l'index car:

```
GET /nom_de_votre_index/_search
{
  "query": {
    "match_all": {}
  }
}
```

- Supprimer le document:

```
DELETE /car/_doc/1?pretty
```



# Le traitement par lots:

- Pour mettre à jour plusieurs documents en utilisant l'API **\_bulk**

POST /car/\_bulk?pretty

```
{"index":{"_id":"1"}}
```

```
{"name": "Tata Nexon" }
```

```
{"index":{"_id":"2"}}
```

```
{"name": "Tata Nano" }
```

# Agrégation:


## ► Agrégation:

GET /index/\_search

```
{  
  "aggs" : {  
    "avg_value" : {  
      "avg" : {  
        "field" : "name_of_field"  
      }  
    }  
  }  
}
```

# Testons...

- ▶ Nous avons une base de données relationnelles BDEmp avec la table employes  
Employes( id, nom, age, email, date\_embauche).
- ▶ Nous souhaitons
  - Créer un index sous Elasticsearch avec un mapping explicite
  - Insérer un employé avec la méthode post avec un index implicite ( non spécifié)
  - Insérer 3 employés en utilisant bulk
  - Afficher tous les employés.
  - Modifier le mapping en rajoutant un champ salaire de type float
  - Insérer des salaires pour tous les employés ( 2200, 3150, 2913.50, 3409)

- 
- Chercher les employés qui ont un nom contenant « Doe » ( exemple d'une partie d'un nom inséré).
  - Cherchez les employés qui ont un email gmail
  - Afficher les noms et emails de tous les employés
  - Chercher les employés qui ont un salaire >3000 et n'ayant pas d'adresse gmail.
  - Chercher les employés qui ont une date d'embauche ultérieure à 31-12-2023 ou ceux qui ont un âge entre 25 et 35.
  - Calculer la moyenne des salaires des employés.
  - Calculer la moyenne des salaires dont l'âge >40 ( 40 par exemple, adaptez ce nombre)
  - Calculer le nombre d'employés qui ont un age >25
  - Supprimer un employé qui a une date d'embauche avant une date que vous fixez par rapport à votre jeu de données.

# Bibliographie

- ▶ [1] Advanced Elasticsearch 7.0, Packt edition, par Wai Tak Wong
- ▶ [2] <https://riptutorial.com/fr/home>