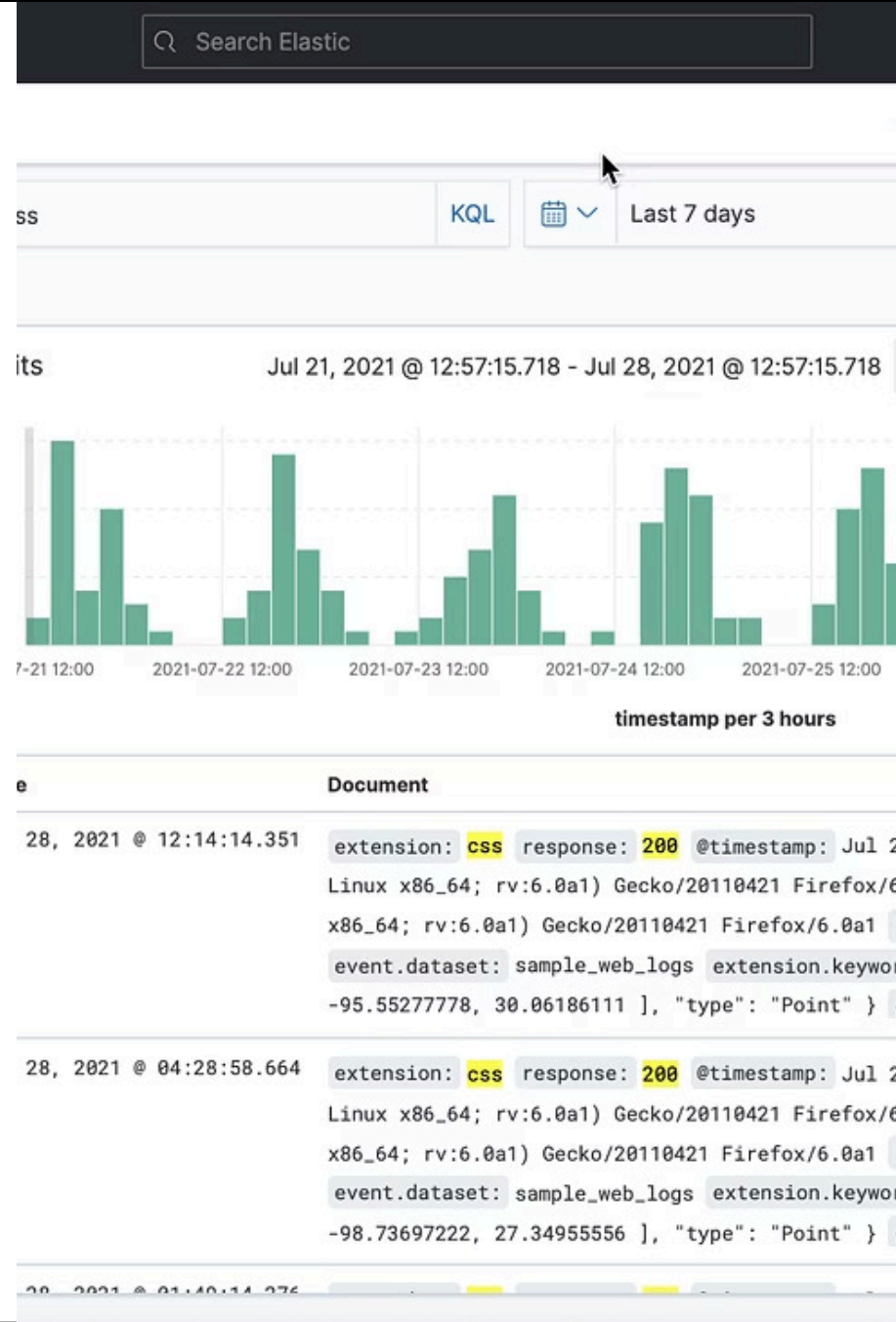


# Kibana Query Language



# Où utiliser KQL dans Kibana ?

## Discover

Explorez vos documents bruts et testez vos requêtes KQL en temps réel

## Lens

Créez des visualisations interactives en filtrant les données avec KQL

## Dashboards

Appliquez des filtres globaux ou locaux pour affiner vos tableaux de bord

## Visualize

Construisez des graphiques personnalisés basés sur des requêtes KQL précises

# KQL vs DSL vs ES|QL : quelle différence ?



## KQL (Kibana Query Language)

### Interface utilisateur simple

Syntaxe intuitive et lisible pour filtrer rapidement dans Kibana. Idéal pour les recherches courantes et les utilisateurs non techniques.

*Exemple* : `customer_gender : "MALE" and price > 50`



## DSL (Domain Specific Language)

### Requêtes JSON avancées

Langage de requête complet d'Elasticsearch en JSON. Offre toute la puissance du moteur mais nécessite une syntaxe plus complexe.

*Exemple* : Format JSON avec match, bool, aggregations

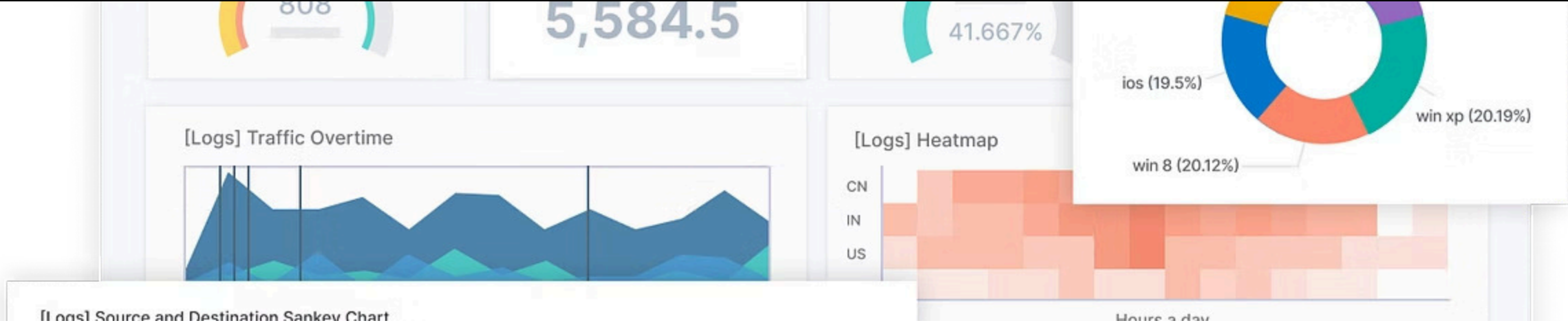


## ES|QL (Elasticsearch Query Language)

### Langage type SQL

Nouveau langage inspiré de SQL pour des analyses complexes. Combine simplicité et puissance pour les requêtes analytiques avancées.

*Exemple* : `FROM logs | WHERE status == 200`



# Chapitre 1

## Les bases du langage KQL

# Structure d'une requête KQL

## Anatomie d'une requête

Une requête KQL se compose de trois éléments fondamentaux qui forment la base de toute recherche :

- **Champ** : le nom du champ à interroger
- **Opérateur** : définit la relation (égalité, comparaison)
- **Valeur** : la donnée recherchée

## Syntaxe de base

```
champ : "valeur"
```

## Exemples pratiques

```
customer_gender : "MALE"  
category : "Men's Clothing"  
email : *gmail.com
```

La syntaxe est intuitive : le champ, suivi de deux-points, puis la valeur recherchée entre guillemets.

# Types de recherches simples

## Égalité exacte

Recherche une valeur précise dans un champ

```
status : "active"
```

Trouve tous les documents où le statut est exactement "active"

## Recherche par préfixe

Utilise le wildcard \* pour les correspondances partielles

```
email : *gmail.com
```

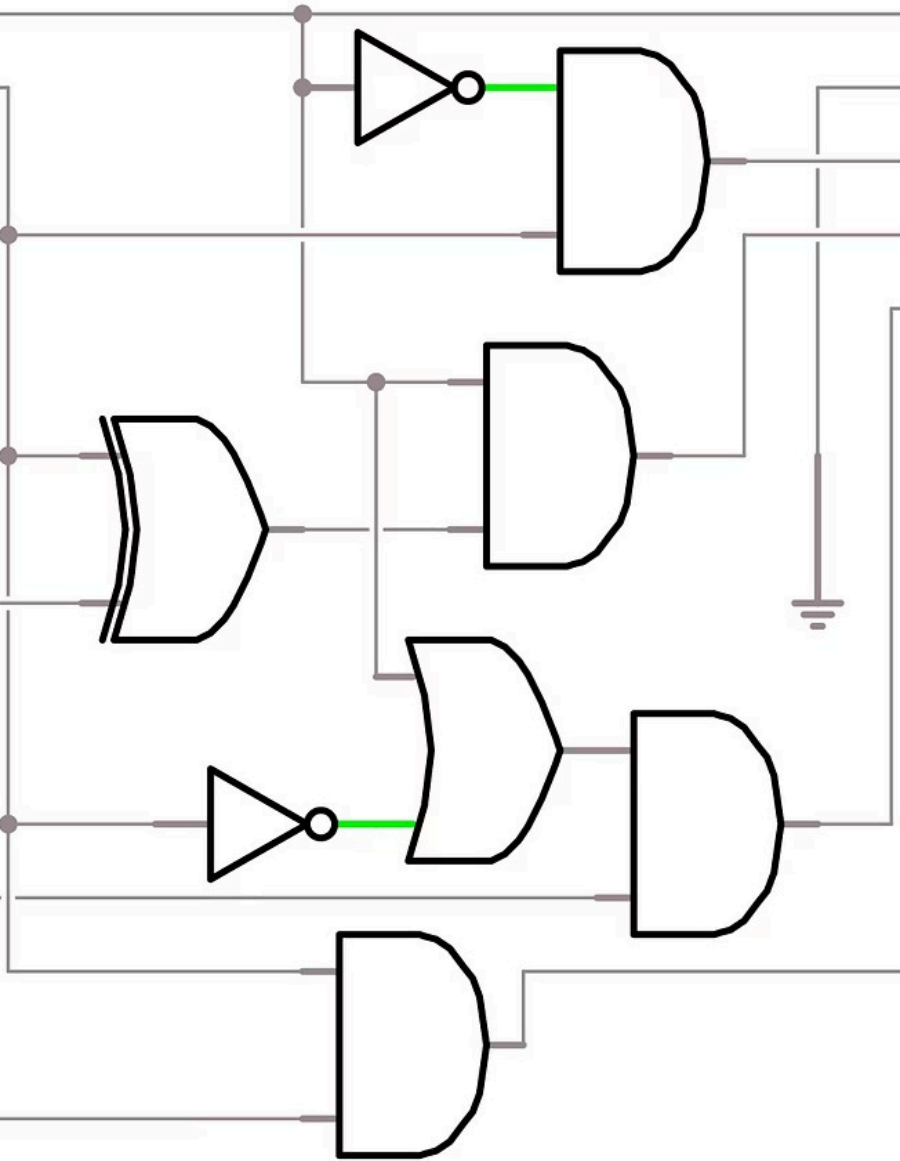
Trouve toutes les adresses se terminant par gmail.com

## Insensible à la casse

KQL ignore automatiquement la casse pour les champs text

```
city : "paris"
```

Trouvera "Paris", "PARIS" ou "paris" indifféremment



# Chapitre 2

## Opérateurs logiques

# Combiner vos conditions avec précision



## AND

Les deux conditions doivent être vraies

```
gender : "MALE" and price > 50
```



## OR

Au moins une condition doit être vraie

```
category : "Shoes" or category :  
"Boots"
```



## NOT

Exclut les documents correspondants

```
not currency : "EUR"
```

---

## Priorité et parenthèses

Utilisez des parenthèses pour contrôler l'ordre d'évaluation des conditions complexes :

```
(customer_gender : "MALE" and taxful_total_price > 50) or category : "Electronics"
```



# Bonnes pratiques pour les opérateurs logiques



## Lisibilité avant tout

Utilisez des parenthèses même quand ce n'est pas strictement nécessaire pour clarifier vos intentions et faciliter la maintenance



## Optimisez les performances

Placez les conditions les plus restrictives en premier pour réduire le volume de données à traiter



## Testez progressivement

Construisez vos requêtes complexes étape par étape en validant chaque ajout de condition dans Discover

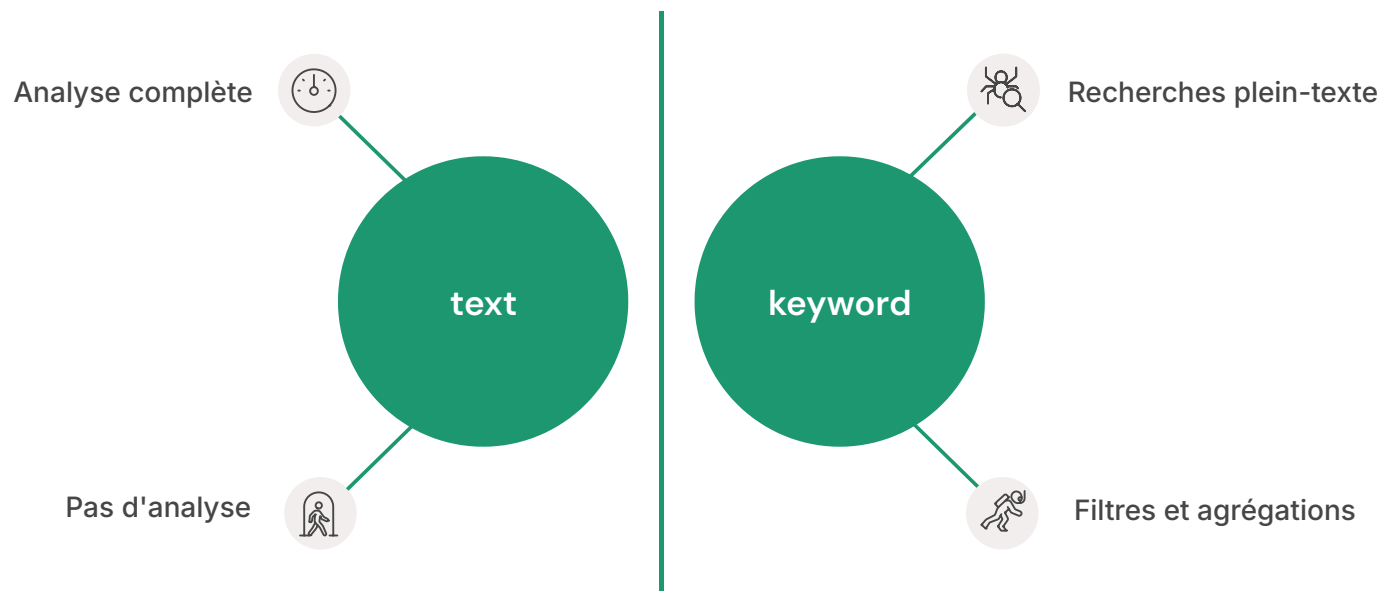


# Chapitre 3

## Recherches textuelles

# Champs Texte dans Elasticsearch et KQL

Pour des recherches textuelles efficaces dans Elasticsearch, il est crucial de comprendre la distinction entre les types de champs `text` et `keyword`. Cette distinction influence directement la manière dont vos données sont indexées, et par conséquent, comment KQL (Kibana Query Language) les interprète et les recherche.



# Comprendre l'Impact sur KQL avec des Exemples Pratiques

## Exemple avec un champ text

Supposons un champ `description_produit` de type `text` contenant la valeur :  
"Smartphone haute performance avec écran OLED".

Lorsqu'un champ est de type `text`, Elasticsearch effectue une analyse (tokenisation, minuscule, suppression des mots vides, stemming, etc.). Par exemple, "performance" pourrait être indexé comme "perform".

### Requête KQL :

```
description_produit:performant
```

Cette requête KQL trouverait la phrase "haute performance" car le terme "performant" serait analysé en "perform" (stemming), correspondant au terme "performance" également stemmisé en "perform" dans l'index.

```
description_produit:"haute performance"
```

Cette requête trouverait la correspondance exacte de la phrase analysée.

## Exemple avec un champ keyword

Supposons un champ `code_produit` de type `keyword` contenant la valeur :  
"SMARTPHONE-HP-OLED-2023".

Lorsqu'un champ est de type `keyword`, la valeur est indexée telle quelle, sans aucune analyse. C'est une correspondance exacte.

### Requête KQL :

```
code_produit:SMARTPHONE
```

Cette requête ne trouverait pas la valeur "SMARTPHONE-HP-OLED-2023" car "SMARTPHONE" n'est pas une correspondance exacte de toute la chaîne indexée.

```
code_produit:SMARTPHONE-HP-OLED-2023
```

Cette requête trouverait la valeur, car elle correspond exactement à la chaîne indexée.

```
code_produit:SMARTPHONE*
```

Cette requête utiliserait une recherche par joker (wildcard) pour trouver les valeurs commençant par "SMARTPHONE", ce qui fonctionnerait avec le champ `keyword`.

# Wildcards et recherches partielles



## Wildcard \* (zéro ou plusieurs)

Remplace n'importe quelle séquence de caractères

```
email : *@gmail.com  
manufacturer : "Low*Media"
```

Trouve tous les emails Gmail ou les fabricants commençant par "Low" et finissant par "Media"



## Wildcard ? (un caractère)

Remplace exactement un caractère unique

```
product_code : "AB?123"
```

Trouve AB1123, AB2123, ABX123, etc.




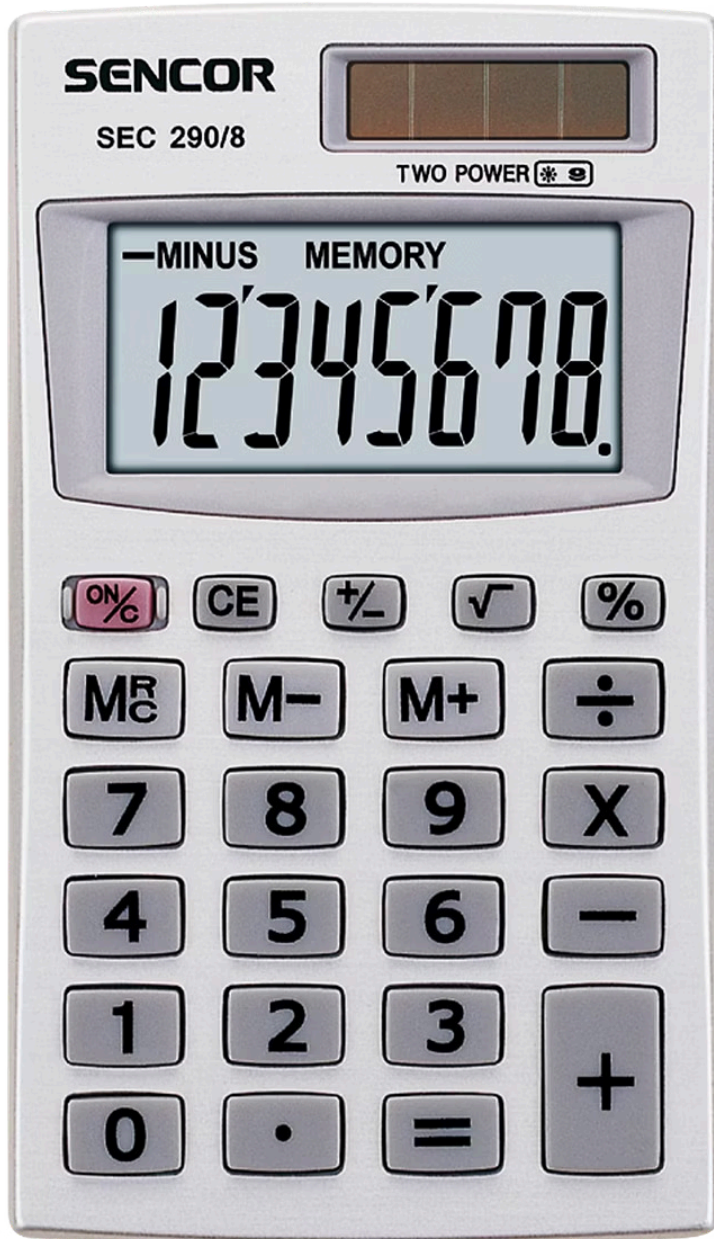
## Recherche exacte

Utilisez des guillemets pour les correspondances précises

```
products.product_name.keyword :  
"Sweatshirt - navy"
```

Trouve uniquement cette phrase exacte, respectant espaces et ponctuation

 **Attention :** Les wildcards au début d'un mot (\*valeur) sont coûteux en performance. Utilisez-les avec parcimonie !



# Chapitre 4

## Recherches numériques

# Opérateurs de comparaison

&gt;

Supérieur >

price > 100

&lt;

Inférieur <

quantity < 5

&gt;=

Supérieur ou égal >=

total >= 50

&lt;=

Inférieur ou égal <=

discount <= 20

---

## Plages de valeurs avec TO

Utilisez la syntaxe [min TO max] pour définir des intervalles numériques inclusifs :

taxful\_total\_price : [20 TO 100]

Trouve toutes les transactions entre 20 et 100 (inclus)

total\_quantity : [2 TO 5]

Filtre les commandes avec 2 à 5 articles

# Combiner conditions numériques et logiques

Les requêtes numériques deviennent puissantes quand on les combine avec des opérateurs logiques :

```
taxful_total_price >= 20 and taxful_total_price <= 100 and customer_gender : "FEMALE"
```

Cette requête trouve toutes les clientes ayant effectué des achats dans une fourchette de prix spécifique.

```
(total_quantity > 3 or taxful_total_price > 200) and currency : "USD"
```

Identifie les grosses commandes (par quantité ou montant) en dollars américains.





# Chapitre 5

## Filtres temporels

# Gérer les dates avec KQL

## Dates absolues

Spécifiez des dates exactes au format ISO

```
order_date >= "2025-01-01"
```

1

2

## Dates relatives

Utilisez `now` pour des périodes dynamiques

```
order_date >= now-7d
```

## Plages temporelles

Combinez dates avec TO pour des intervalles

```
order_date : [now-30d TO now]
```

3

4

## Arrondissement

Utilisez `/d`, `/w`, `/M` pour arrondir

```
order_date >= now/d
```

# Expressions temporelles courantes

## Derniers 7 jours

```
order_date >= now-7d
```

Données de la semaine écoulée

## Dernier mois

```
order_date >= now-30d
```

Transactions des 30 derniers jours

## Aujourd'hui uniquement


```
order_date >= now/d
```

Commandes du jour en cours

## Année en cours

```
order_date >= now/y
```

Toutes les données depuis janvier

 **Fuseau horaire** : Les dates utilisent le fuseau UTC par défaut. Configurez le fuseau dans les paramètres Kibana si nécessaire.