Elektronski fakultet, Univerzitet u Nišu

Katedra za Računarstvo i informatiku

SEMINARSKI RAD

Obrada upita kod Elasticsearch

Predmet: Sistemi za upravljanje bazama podataka

Student: Aleksandar Mitić

Broj indeksa: 1204

Sadržaj

[Uvod 2](#_Toc69325684)

[Pretraživanje podataka 5](#_Toc69325685)

[Request UI upiti 5](#_Toc69325686)

[DSL upiti 6](#_Toc69325687)

[Kako funkcioniše pretraživanje 7](#_Toc69325688)

[Razumevanje rezultata upita 8](#_Toc69325689)

[Tipovi upita 10](#_Toc69325690)

[Term level upiti 12](#_Toc69325691)

[Full text upiti 16](#_Toc69325692)

[Kreiranje složenih upita pomoću logičkih izraza 18](#_Toc69325693)

[Spojevi (eng. „Joins“) 19](#_Toc69325694)

[Spajanje dokumenata ugnježdavanjem 19](#_Toc69325695)

[Spajanje dokumenata mapiranjem 20](#_Toc69325696)

[Pretraživanje prema identifikacionom broju roditelja 21](#_Toc69325697)

[Pretraživanje dece ukoliko bez identifikacionog broja roditelja 21](#_Toc69325698)

[Pretraživanje roditelja 22](#_Toc69325699)

[Više nivovsko povezivanje (eng. „Multi-level relations“) 23](#_Toc69325700)

[Kontrolisanje rezultata 25](#_Toc69325701)

[Agregacija podataka 28](#_Toc69325702)

[Metrička agregacija 28](#_Toc69325703)

[Bucket agregacija 30](#_Toc69325704)

[Ugnježdena agregacija 31](#_Toc69325705)

[Filtriranje agregacije 31](#_Toc69325706)

[Ručno definisanje uslova za bucket agregaciju 32](#_Toc69325707)

[Opseg agregacija 33](#_Toc69325708)

[Histogrami 33](#_Toc69325709)

[Globalna agregacija 34](#_Toc69325710)

[Agregacija ugnježdenih objekata 35](#_Toc69325711)

[Zaključak 36](#_Toc69325712)

[Literatura 37](#_Toc69325713)

# Uvod

Elasticsearch je besplatan, distribuiran, moderan i analitički alat koji podržava sve tipove podataka, uključujući tekstualne, numeričke, struktuirane i nestruktuirane podatke. Elasticsearch je izgrađen na Apache Lucene platformi [1] i inicijalno je pušten 2010. godine od strane Elasticsearch N.V. (danas poznata kao Elastic) [2]. Poznat po distibuiranoj prirodi, REST API-ju, brzini i skalabilnosti, Elasticsearch predstavlja centralnu komponentu Elastic Stack-a, skupa besplatnih alata za skladištenje, analizu i vizuelizaciju podataka.

Brzina i skalabilnost Elasticsearch-a kao i njegova sposobnost indeksiranja različitih tipova dokumenata znače da se Elasticsearch može iskoristiti za različite upotrebe:

* pretraga aplikacija
* pretraga veb stranica
* evidentiranje u log i analiza loga
* mere infrastrukture
* nadgledanje performasi aplikacije
* analiza i vizuelizacija geoprostornih podataka
* analiza bezbednosti
* poslovna analiza

Tokovi sirovih podataka (eng. „raw data“) se u Elasticsearch slivaju iz različitih izvora. Unos podataka je proces kojim se ovi sirovi podaci raščlanjuju, normalizuju i obogaćuju pre nego što se indeksiraju u Elasticsearch-u. Nakon indeksiranja podataka u Elasticsearch-u, korisnici mogu pokretati složene upite nad njima i primenjivati agregacije za izvlačemnje složenijih zaključavaka. Pomoću Kibane, korisnici mogu da kreiraju moćne vizuelizacije svojih podataka, dele kontrolne table i upravljaju Elastic Stack-om [3].

Indeks kod Elasticsearch-a predstravlja kolekciju dokumenata koji su međusobno povezani. Elasticsearch podatke čuva kao JSON dokumente gde svaki dokument povezuje skup ključeva (imena polja ili svojstva) sa odgovarajućim vrednostima (tekstovi, brojevi, logičke vrednosti, datumi, nizovi vrednosti, geolokacije ili druge vrste podataka). Elasticsearch koristi strukturu podataka koja se naziva obrnuti indeks (eng. „inverted index“), a koja je dizajnirana da omogući vrlo brzo pretraživanje celog teksta. Obrnuti indeks pravi listu od reči koje su jedinstvene i koje se pojavljuju u bilo kom dokumentu i identifikuje sve dokumente u kojima se svaka reč pojavljuje. Tokom procesa indeksiranja, Elasticsearch skladišti dokumente i gradi obrnuti indeks kako bi se podaci dokumenata mogli pretraživati u skoro realnom vremenu. Indeksiranje se pokreće pomoću indeks API-ja, putem kojeg je moguće dodati ili ažurirati JSON dokument u određenom indeksu. Prednosti korišćenja Elasticsearch alata:

* **Elasticsearch je** **brz**. Budući da je Elasticsearch izgrađen na vrhu Apache Lucene-a, on se ističe u pretraživanju celog teksta. Elasticsearch je takođe platforma za pretragu gotovo u realnom vremenu, što znači da je kašnjenje od trenutka indeksiranja dokumenta pa sve dok dokument ne postane dostupan za pretraživanje je vrlo kratko - obično jedna sekunda. Kao rezultat toga, Elasticsearch je vrlo pogodan za vremenski osetljive slučajeve korišćenja kao što su bezbednosna analitika i nadzor infrastrukture.
* **Elasticsearch je po prirodi** **distribuiran**. Dokumenti uskladišteni u Elasticsearch-u se distribuiraju u različitim kontejnerima poznatim pod nazivom šardovi (eng. „shards), koji se dupliraju kako bi se obezbedila redundantost podataka u slučaju kvara hardvera. Distribuirana priroda Elasticsearch-a omogućava mu da se proširi na stotine (ili čak hiljade) servera i da obrađuje petabajte podataka.
* **Elasticsearch dolazi sa** **širokim skupom funkcija**. Pored brzine, skalabilnosti i elastičnosti, Elasticsearch ima niz moćnih ugrađenih funkcija koje čine skladištenje i pretraživanje podataka još efikasnijim, kao što su zbirni podaci i upravljanje životnim ciklusom indeksa.
* **Elastic Stack pojednostavljuje unos podataka, vizuelizaciju i izveštavanje**. Integracija sa Beats-om [4] i Logstash-om [5] olakšava obradu podataka pre nego što se podaci indeksiraju u Elasticsearch-u. Kibana pruža vizuelizaciju Elasticsearch podataka u realnom vremenu, kao i korisničke interfejse za brzi pristup nadzoru performansi aplikacija (APM), evidencijama i podacima metrike infrastructure [6].

# Pretraživanje podataka

Postoje dve metode pisanja upita (eng. „queries“) za pretraživanje podataka kod Elasticsearch-a:

1. pisanje DSL upita
2. request UI upiti

Najfleksibilniji i napogodniji način pisanja upita je pisanje DSL upita. Pisanje ovih upita označava zapravo dodavanje JSON objekta nakon request UI linije. Za pisanje upita se dodaje „query“ objekat koji sadrži definiciju same pretrage podataka, na primer:



**Slika 1**: Primer DSL upita

U primeru na slici 1, „match“ tag predstavlja tip upita koji sadrži ključ sa nazivom polja (kolone) koji se pretražuje. Pored DSL upita request UI upit izgleda ovako:

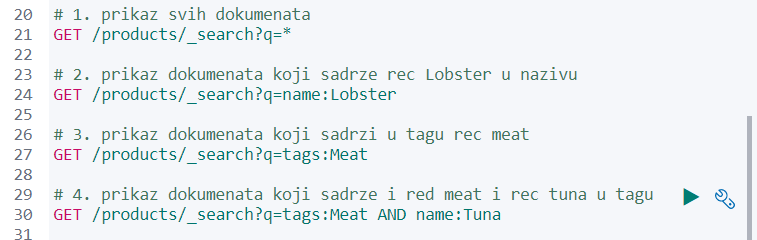


**Slika 2**: Primer request UI upita

Na slici 2, „q“ (query) predstavlja početak upita iza koga se navodi naziv polja koje se pretražuje. U polju „name“ koje je navedeno pretražuje se reč „pasta“. Ovom metodom je moguće pisati veoma složene upite, ali ovaj pristup je manje izražajan od DSL upita jer veoma lako postaje težak za razumevanje i čitanje. Takođe, postoje neke stvari koje jednostavno nije moguće napraviti pomoću ovog pristupa request UI upita, ali je veoma pogodan za pisanje brzih upita poput upita za debagiranje preko komandne linije.

## Request UI upiti

Za pisanje request UI upita na samom početku se šalje GET zahtev (eng. „request“), pa se zatim navodi upit sa parametrom za upit „q“. Na slici 3 su prikazani različiti request UI upiti:



**Slika 3**: Prikaz različitih request UI upita

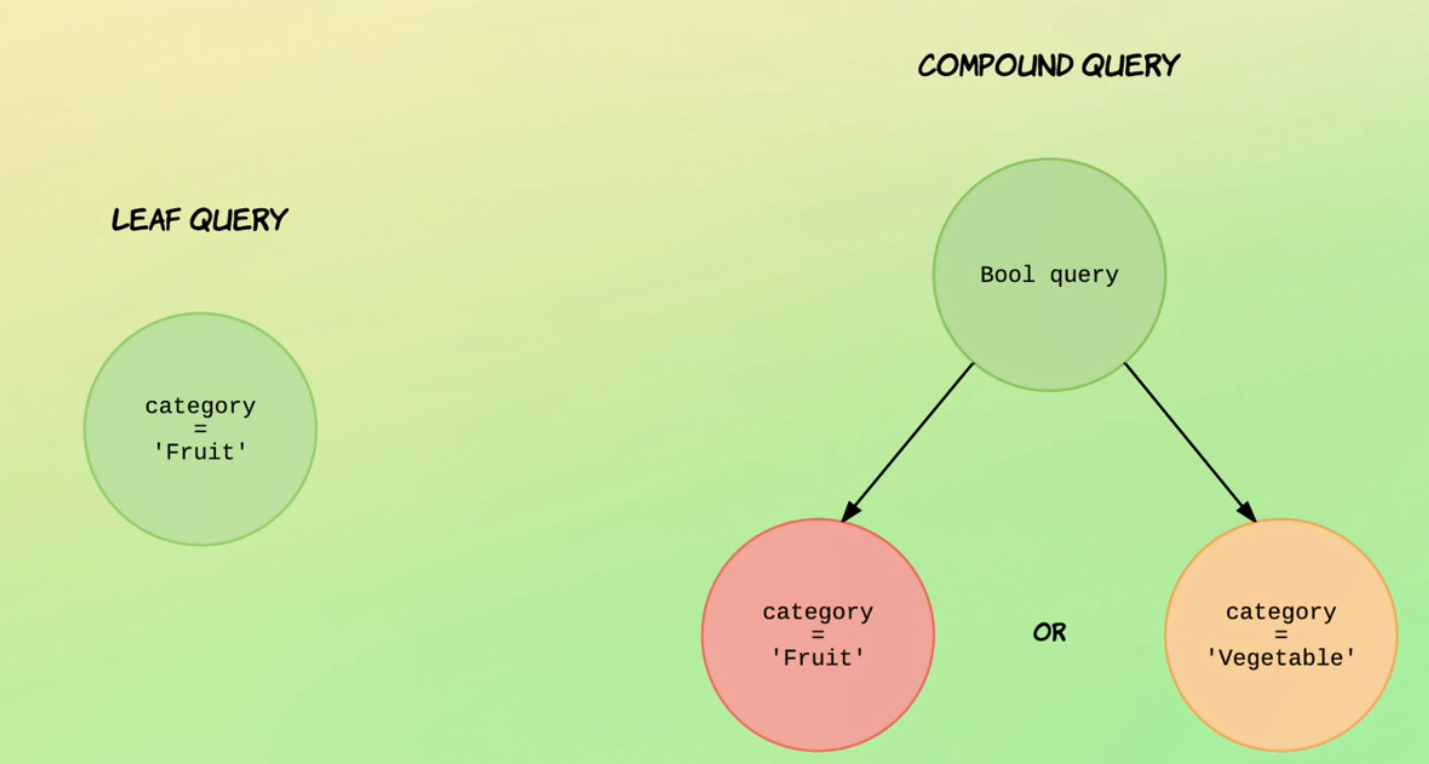
Prvi request UI upit prikazuje sve dokumente iza taga „q“ koji označava početak upita koristi se znak „\*“ za prikaz svih dokumenata (inicijalno Elasticsearch prikazuje samo prvih 10 dokumenata na šta je moguće uticati). Drugi upit vrši prikaz svih dokumenata koji u polju „name“ sadrže reč „Lobster“. Treći upit prikazuje sve dokumente koji u polju „tags“ sadrži reč „Meat“ i četvrti upit prikazuje samo dokumente koji sadrže u polju „tags“ reč „Meat“ i polju „name“ reč „Tuna“.

## DSL upiti

Kao što je već rečeno pisanje DSL upita se realizuje pomoću JSON objekata. Postoje dve glavne grupe upita kod DSL upita:

1. leaf upiti
2. compound ipiti

Leaf (list) upiti se koriste za pretraživanje vrednosti u određenom polju dok se compound upiti sastoje od više leaf upita koji su takođe rekurzivni po prirodi, na primer:



**Slika 4**: Grafički prikaz leaf i compound upita

Leaf upiti se mogu iskoristiti na primer za prikaz dokumenta koji spadaju u kategoriju voća (eng. „Fruit“ na slici 4), dok compound upiti mogu iskombinovati dva leaf upita zajednos sa bool upitom da bi se prikazali dokumenti koji pripadaju kategoriji „Fruit“ ili kategoriji „Vegetable“ (primer sa slike 4). DSL upit za prikaz svih dokumenata je dat na slici 6:



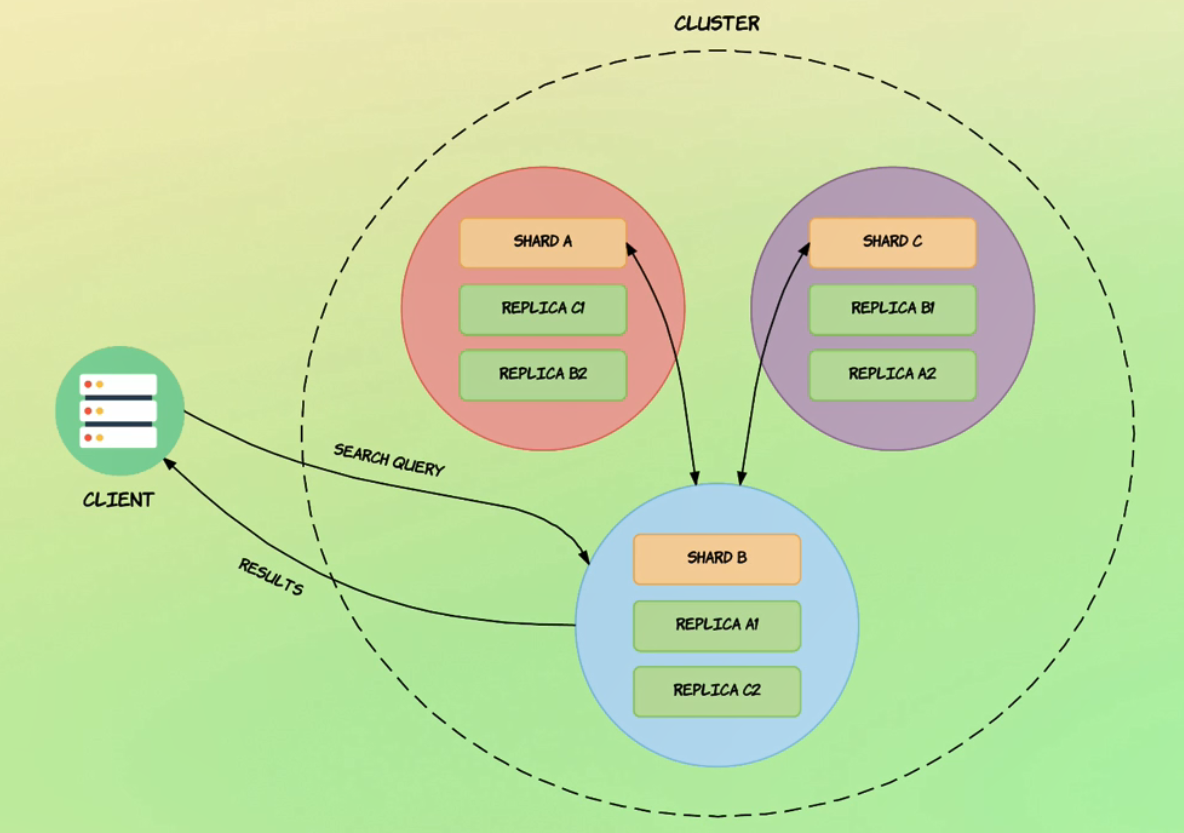
**Slika 6**: Prikaz svih dokumenata korišćenjem DSL upita

## Kako funkcioniše pretraživanje



**Slika 7**: Grafički prikaz pretraživanja u Elasticsearh-u

Na slici 7, sa leve strane je klijent koji je uglavnom predstavlja server, koji komunicira sa Elasticsearch klasterom šaljući mu http zahtev za pretraživanje. Klaster zatim izvodi svoju magiju zasnovanu na indeksima i upitu koji je prosleđen preko http zahteva. Kada je rezultat gotov, klaster odgovara slanjem dobijenih rezultata klijentu, koje klijent može dodatno koristiti. Pretpostavimo da klaster sadrži tri čvora sa jednim indeksom koji je distriburiran preko tri šarda A, B i C:



Slika 8: Primer slanja zahteva u klasteru sa 3 čvora

Svaki šard sadrži po dve replike, tako da se svaka replikaciona grupa sasoji od primarnog šarda i dve replike. Pretpostavimo da klijent pošalje upit klasteru koji se prosleđuje čvoru koji sadrži šard B. Ovaj čvor se naziva koordinator („coordinated node“) čime se označava da je on odgovoran za slanje upita drugim čvorovima u klasteru, prihvati rezultat i da ga pošalje klijentu. Inicijalno, svaki čvor može da se ponaša kao coordinator to znači da svaki čvor može da prihvati HTTP zahtev. Kako koordinacioni čvor sadrži šard koji je potrebno pretražiti čvor će sam izvršiti upit. Naravno, ovo ne mora uvek da bude uvek slučaj ali kako se ovde radi sa jednim indeksom ovo će uvek biti slučaj. Koordinacioni čvor zatim šalje (eng. „broadcast”) zahtev svakom šardu sa tim indeksom nebitno da li čvorovi sadrže primarni šard ili repliku. Čvorovi zatim šalju koridinacionom čvoru rezultate koje on spaja i sortira i takav rezltat vraća nazad klijentu.

## Razumevanje rezultata upita

Nakon pokretanja upita dobija se rezultat koji izgleda ovako:



**Slika 9**: Prikaz rezultata dobijenog pozivom upita

Na slici 9, u drugoj liniji tag „took“ označava koliko mislisekundi je potrebno da se upit izvrši. Tag „time\_out“ u trećoj liniji predstavlja fleg koji označava da li je došlo do time out exception ili nije. Tag „\_shards“ sadrži ukupan broj šardova „total“ koji su pretraživani kao i broj štardova koji su završeni uspešno (successful), koji su preskočeni (skipped) i koji nisu pretraženi (failed). Bitnije informacije, unutar taga „hits“ se nalazi informacija o broju vraćenih dokumenata koji odgovaraju kriterijumima pretrage (1000 dokumenta u ovom primeru). Tag „hits“ u 16. liniji predstavlja niz koji sadrži dokument koji odgovara kriterijumima pretrage. Svaki dokument sadrži „\_score“ atribut, kojim se označava koliko dokument zapravo odgovra kritrerijumima pretrage. Atribut „max\_score“ u 15. liniji čuva vrednost najvećeg skora koji odgovara dokumetnu koji je vraćen upitom.

# Tipovi upita

Kod Elasticsearch-a postoje 2 gupe upita „full text“ upitii „term level“ upiti. Primer ova dva upita je dat na slici 10:



**Slika 10**: Primer full text queries i term level queries

Prva dva upita se odnose na grupu „term level“ upite jer sadrže atribut „term“ iza atributa „query“. Ova upita pokušavaju da vrate proizvod koji u imenu sadrži naziv „lobster“ s tim da se u drugom upitu prosleđuje „Lobster“ sa prvim velikim slovom. Kada se pokrene prvi upit sa malim prvim slovom u upitu, upit vraća dva dokumenta od kojih oba dokumenta sadrže „Lobster“ sa prvim velikim slovom, dok drugi upit ne vrati ništa:

**Slika 13**: Rezultat prvog „term level” upita

**Slika 14**: Rezultat drugog „term level” upita

 Iako se sa slike 13 vidi da u bazi postoje 2 dokumenta (atribut „value“ označen crvenom bojom) koja sadrže naziv „Lobster“ sa velikim počtenim slovom (atribut „name“), upit sa velikim početnim slovom ne vraća ništa. Razlog za jesu invertovani indeksi kod Elasticsearch-a gde se podatci konvertuju u mala slova i tako se skladište u bazi. Dakle, kod prvog upita dolazi do sledeće situacije kao na slici 15:



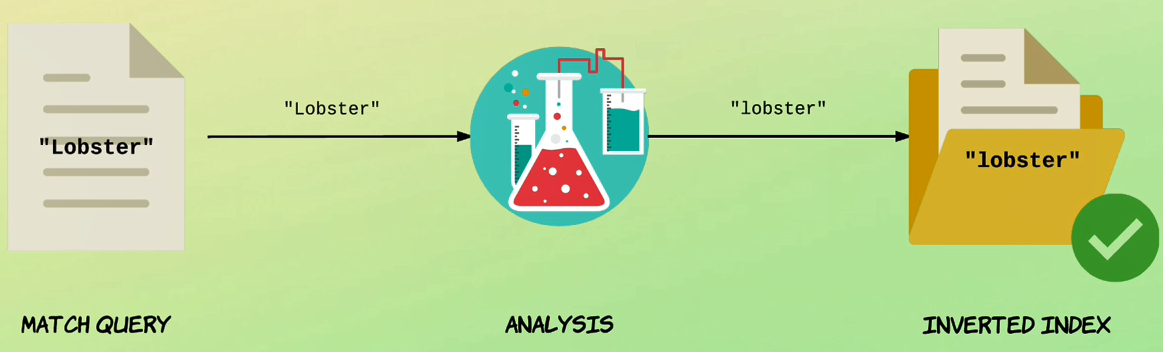
**Slika 15**: Prvi „term level” upit u pozadini

Elasticsearch pokušava da pronađe u nazivu „lobster“ i pronalazi ga, podaci u bazi su zapisani malim slovima. Dok, s druge strane Elasticsearch pokušava da pronađe reč „Lobster“ u nazivu i ne uspeva, jer su podaci u bazi konvertovani i uskladišteni sa malim slovima u nazivu, primer na slici 16:



**Slika 16**: Drugi „term level” upit u pozadini

Treći „full text“ upit vraća oba dokumenta kao i prvi „term level“ upit, razlog za to jeste što kod „full text“ upita postoji među faza gde se upit konvertuje svako veliko slovo u malo, pa se tek onda vrši pretraživanje. Na slici 17 prikazan je „full text“ upit:



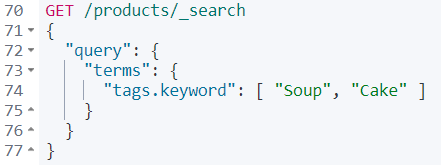
**Slika 17**: Treći „full text” upit u pozadini

U prevodu, „term level“ upiti traže tačno pokalapanje vrednosti u upitu i u bazi dok i ne analiziraju se kao što je to slučaj kod „full text“ upita. Naravno ovo sve važi ukoliko se koristi standardni analizator kod Elasticsearch-a koji vrši ovu konverziju. U suprotnom ukoliko se koristi neki drugi analizator koji ne izvršava ovu konverziju „term level“ upiti mogu da vrate null vrednost. Zbog toga je pogodno koristiti „full text“ upite kako se dobili rezultati koji se očekuju.

## Term level upiti

Term level upiti kao što je već pomenuto se koriste uglavnom za pretraživanje konkretnih vrednosti, pogodni su kod pretraživanja datumskih polja, tekstualnih polja kao i kod pretrživanja brojeva i logičkih polja. Primeri term level upita:

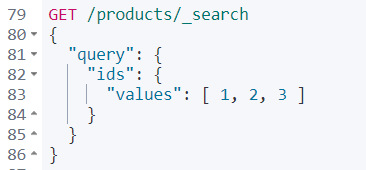
1. pretraživanje više različitih vrednosti u nizu „tags“ potrebno je navesti ključnu reč „keywords“ i navesti listu vrednosti za pretragu kao na slici 18, gde se pretražuju tagovi „Soup“ i „Cake“:



**Slika 18**: Term level pretraživanje više vrednosti različitih vrednosti

Inicijalni operator koji se koristi je „ili“ (eng. „OR“) tako da upit sa slike 18 vraća sve dokumente koje sadrže bar jedan od ova dva taga, ali kao prvi dokument sa najvećim skorom vraća onaj koji sadrži oba taga (naravno ukoliko postoji takav dokument).

1. pretraživanje dokumenata prema unapred poznatom id-ju dokumenta. Na slici 19 je prikazano čitanje dokumenata sa id-jem 1,2 i 3:



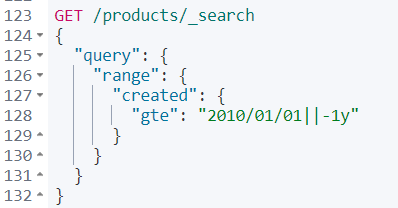
**Slika 19**: Term level pretraživanje dokumenata prema id-ju

1. pretraživanje dokumenata prema opsegu vrednosti. Na slici 20 su data 3 različita term level upita za pretraživanje dokumenata u opsegu vrednosti. Prvim upitom se pretražuju svi dokumenti čiji je atribut „in\_stock“ veći ili jednak od 1 (na slici 20 označeno atributom „gte“) i manji ili jednak od 5 (na slici 20 označeno atributom „lte“). Drugim i trećim upitom se traže dokumenti koji su kreirani 2010. godine s tim da treći term level upit ima drugačiji format datuma koji je specificiran atributom „format“:



**Slika 20**: Term level pretraživanje dokumenata u opsegu vrednosti

1. pretraživanje dokumenata sa relativnim datumom. U pitanju je modifikacija pretraživanja dokumenata u opsegu dodavanjem znaka „| |“ iza koga sledi matematički izraz kao na slici 21. Matematički izraz koji je dodat u primeru prikazanom na slici 21 jeste oduzimanje godine od datuma 01.01.2010. (oduzimanje je označeno sa -1y gde je y oznaka za godinu eng. „year“). Tako da se u ovom slučaju pretražuju dokumenti koji su kreirani nakon 01.01.2019. [7]:



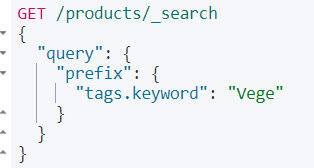
**Slika 21**. Term level pretraživanje sa relativnim datumom

1. pretraživanje dokumenata sa „non-null” vrednostima se realizuje pomoću atributa „exists” u upitu prikazanom na slici 22:



**Slika 22**: Term level pretraživanje dokumenata koji sadrže vrednosti u nizu „tags”

1. pretraživanje dokumenata koji počinju sa prefiksom. Ovakvi term level upiti se realizuju pomoću atributa „prefix“ u upitu. Na slici 23 je prikazan term level upit koji pretražuje dokumente koji u nizu „tags“ počinju sa prefiksom „Vege“ (skraćeno od eng. „Vegetables“):



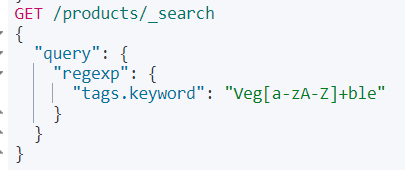
**Slika 23**: Term level pretraživanje pomoću prefiksa

1. pretraživanje pomoću „wildcard“ znakova. Wildcard znakovi kod Elasticsearch-a su „\*“ i „?“. Znak „\*“ označava više karaktera dok znak „?“ označava tačno jedan karakter. Na slici 24 dati su primeri korišćenja wildcard znakova. Prvim upitom se pretražuju dokumenti koji sadrže u nizu „tags“ vrednost koja počinje sa „Vege“ i završava se sa „ble“ korišćenjem „\*“ karaktera. U drugom upitu se traže vrednosti koje počinju sa „Veget“ i završavaju se sa „ble“ dok se izmežu ova dva teksta može nalaziti bilo koji karakter.



**Slika 24**: Term level pretraživanje pomoću wildcard karaktera

1. pretraživanje regularnih izraza [8]. Primer term level pretraživanja pomoću reqularnih izraza dat je na slici 25 gde se pretražuju vrednosti u nizu „tags“ koji sadrže samo mala i velika slova između „Veg“ i „ble“:



**Slika 25**: Term level pretraživanje pomoću reqularnih izraza

## Full text upiti

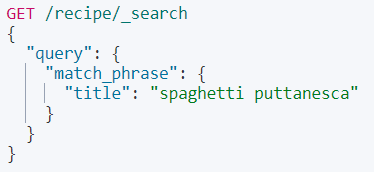
Za pretraživanje teksta pogodniji su full text upiti. Inicijalni operator za pretragu je „ili“ (eng. „OR“). Na slici 26 prikazan je primer pretraživanja recepata sa inicijalnim operatorom „ili“ i sa operatotom „i“ (eng. „and“). Razlika između „i“ i „ili“ operatora je da kod „i“ operata moraju se pojaviti sve vrednosti definisane u upitu dok se kod „ili“ operatora pojavljuju dokumenti sa bar jednom specificiranom vrednosšću u upitu.



**Slika 26**: Full tekst pretraživanje sa „i” i „ili” operatorom

Kod prvog upita na slici na slici 26, pretražuju se dokumenti sa „ili“ operatorom tj. pretražuju se dokumenti koji sadrže bar jednu reč iz uslova „Recipes with pasta or spaghetti“. Naravno dokument sadrži sve reči u naslovu onda će taj dokument imati najveću vrednost u „score“ atributu. Kod drugog upita se pretražuju dokumenti sa „i“ operatorom gde dokumenti moraju da sadrže sve tri reči „pasta or spaghetti“. Ovo pretraživanje je veoma slično term level pretraživanju jer se traže konkretne vrednosti.

Pomoću full text pretraživanja moguće je pretraživati i fraze u dokumentu pomoću atributa „match\_phrase“ iza koga se navodi polje kao i vrednost fraze koja se pretražuje u navedenom polju. Na slici 27 prikazan primer full text pretraživanja fraze „spaghetti puttanesca“. Ovo pretraživanje nije isto ukoliko bi se rečima zamenila mesta „puttanesca spaghetti“, dakle kod pretraživanja pomoću fraze veoma je bitan redosled u teksta u upitu.



**Slika 27**: Full text pretraživanje fraza

Takođe, moguće je pretraživati više različitih polja pomoću atributa „multi\_match” [9]. Na slici 28 prikazan je primer pretraživanja reči „pasta“ u poljima „title“ i „description“:



**Slika 28**: Full text pretraživanje više polja

# Kreiranje složenih upita pomoću logičkih izraza

Kod DSL upita su objašnjeni kompleksiji (eng. „compound”) upiti koji se kreiraju pomoću atributa „bool“. Za kreiranje složenog upita pomoću „bool“ izraza prikazanog na slici 29, iskorišćeni su operatori „must“, „must\_not“, „should“ i „filter“. Atribut „must“, koji je u prethodnim poglavljima prikazan, ukazuje na upit koji se mora pojavljivati u rezultatu. Atribut „must\_not“ ukazuje na upit koji ne sme da se nalazi u rezultatu, tekst definisan pod „should“ atributom može a i ne mora se pojavljivati u rezultatu. Ukoliko se pojavljuje onda će ovaj atribut znatno uvećati „\_score“ atribut tog dokumenta u rezultatu. Atribut „filter“ se koristi za filtriranje dokumenata odnosno za sužavanje rezultata. Na slici 29 je prikazan upit koji traži recepte koji se prave sa parmezanom (definisan pod atributom „must“), koji ne sadrže tunjevinu (definisan pod atributom „must\_not“), poželjno je da sadrži peršun (definisan pod atributom „should“) i da je vreme spremanja tog recepta ispod 15 minuta (definisan pod atributom „filter“):



**Slika 29**: Kreiranje složenog upita bool izrazima

# Spojevi (eng. „Joins“)

Suprotno od relacionih baza podataka, spajanje tabela kod Elasticsearch-a se razlikuje tj. podaci se denormalizuju kad god je to moguće jer to vodi ga boljim performansama. Zbog same denormalizacije podataka, Elaticsearch se ne preporučuje da se koristi kao glavna baza podataka za skladištenje podataka jer denormalizacija podataka utiče na prostor na disku. Ukoliko se Elasticserach ne koristi kao glavna baza podataka, onda je moguće podatke skladištiti po slobodnom izboru da bi se optimizovalo pretraživanje podataka. U prevodu, Elasticsearch ne pordžava spojeve (eng. „joins“) tabela kao što podržavaju relacione baze podataka, ali podržava neke proste načine spajanja dokumenata. Treba imati u vidu da ova spajanja kod Elasticsearch-a mogu znatno uticati na performanse pretraživanja ukoliko se radi sa velikom količinom podataka.

## Spajanje dokumenata ugnježdavanjem

Korišćenjem atributa „nested“ u upitu moguće je spojiti više dokumenata u jedan dokument. Na primer spajanje dokumenata vezom N:1 kao na slici 30 gde su u dokumentu „department“ kreirana dva ugnježdena objekta „employees“ (radnici) koji su tima „nested“:



**Slika 30**: Primer spajanja ugnježdavanjem

Ukoliko se dokumenti spoje na ovaj način njihovo pretraživanje je moguće samo pomoću „nested” atributa u upitu kome se navodi kao putanja (eng. „path”) naziv ugnježdenog atributa koji se pretražuje kao na slici 31 označeno crvenom bojom:



**Slika 31**: Pretaživanje ugnježdenih objekata u dokumentu

## Spajanje dokumenata mapiranjem

Mapiranjem je takođe moguće povezati dokumente jednostavnim navođenjem „join“ kao tip spajanja u upitu i navođenjem relacije tj. veze za spajanje objekata pod atributom „relations“. Na slici 32, prikazano je spajanje objekata „department“ i „employee“ mapiranjem:



**Slika 32**: Primer spajanja dokumenata mapiranjem

Dodavanje dokumenata za ovakvo mapiranje se razlikuje od klasičnog dodavanja dokumenata. Definisanjem relacije, kod dodavanja novih podataka mora se naglasiti polje za document koji se dodaju nove vrednosti pod atributom „join\_field”. Dodavanjem novih departmana potrebno je navesti za „join\_field“ : „department“ (označeno plavom bojom na slici 33) kako bi Elasticsearch znao da se ti podaci odnose na departmane (vrednost „department“ je zapravo vrednost iz atributa „relations“ sa slike 32). Isto tako za dodavanje novih zaposlenih potrebno je pored navođenja „employee“ za „join\_field“ potrebno je navesti i identifikacioni broj dokumenta za koji se dodaje taj zaposleni pod atributom „parent“ (označeno crvenom bojom) kao na slici 33:



**Slika 33**: Dodavanje novih dokumenata odgovarajućim mapiranjem

## Pretraživanje prema identifikacionom broju roditelja

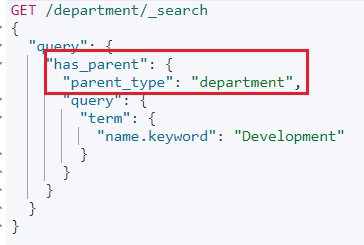
Najjednostavnije spajanje dokumenata je spajanje preko identifikacionog broja roditelja (eng. „id“). Primer dat na slici 34 pretražuje sve zaposlene (eng. „employee”) koji sadrže vrednost identifikacionog broja roditelja 1:



**Slika 34**: Pretraživanje pomoću identifikacionog broja

## Pretraživanje dece ukoliko bez identifikacionog broja roditelja

Uglavnom identifikacioni broj roditelja nije poznat, u tom slučaju moguće je pretraživati decu prema drugim kriterijumima [11]. Potrebno je samo definisati da pretraživani podatak sadrži roditelja u atributu „has\_parent“ i navesti tip roditelja u atributu „parent\_type“ (označeno crvenom bojom na slici 35). Na slici 35 je dat primer pretraživanja dece bez identifikacionog broja:



**Slika 35**: Pretraživanje dokumenata bez identifikacionog broja roditelja

## Pretraživanje roditelja

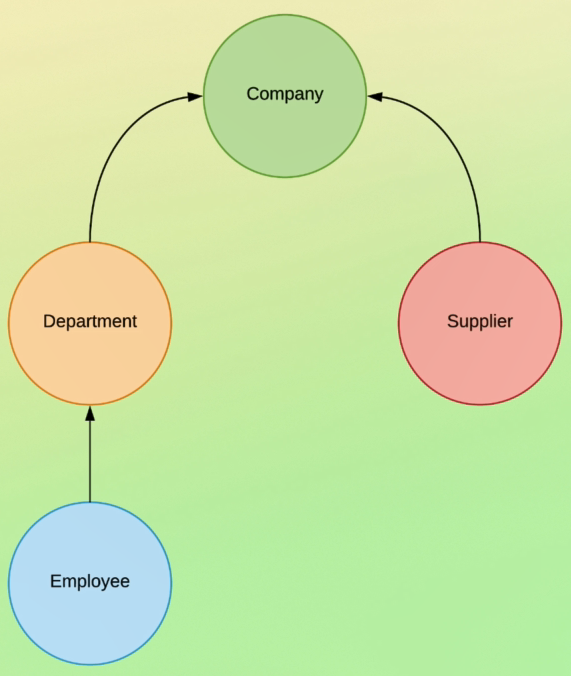
Obrnuta procedura pretraživanja roditelja na osnovu dece je slična, jedina razlika jeste u nazivu atributa, umesto atributa „has\_parent” se koristi atribut „has\_child“ [12]. Na slici 36, dat je primer pretraživanja departmana (roditelja) koji imaju muške zaposlenike (decu) starije od 50 godina:



**Slika 36**: Pretraživanje roditelja

## Više nivovsko povezivanje (eng. „Multi-level relations“)

Kod Elasticsearch-a je moguće i više nivovsko povezivanje objekata tj. višenivovsko ugnježdavanje. Za demonstraciju višenivovskog povezivanja kreiran je dokument kompanija (eng. „company“) sadrži više departmana (eng. „departmant“) i više dobavljača (eng. „supplier“) gde svaki departman sadrži više radnika (eng. „employee“) što je prikazano na slici 37:



**Slika 37**: Primer hijerarhije višenivovskog povezivanja kod Elasticsearch-a

Višenivovsko povezivanje dokumenata je moguće realizovati mapiranjem na isti način kao i kod jedninivovskog povezivanja dokumenata koje je objašnjeno u ovom poglavlju. Jedina razlika je dodavanje novih realacija koje su prikazane na slici 38 (označeno crvenom bojom):



**Slika 38**: Primer višenivovskog mapiranja kod Elasticsearch-a

Na slici 38, crvenom bojom su prikazane opisane relacije. Kompanija sadrži departmane i dobavljače dok departmani sadrže zaposlene. Za pretraživanje zaposlenih potrebno je u upitu navesti dva puta atribut „has\_child“ kao na primeru prikazanom na slici 39 gde se traži zaposleni „John Doe“:

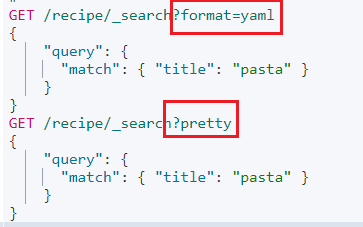


**Slika 38**: Primer pretraživanja višeninovskog mapiranja

# Kontrolisanje rezultata

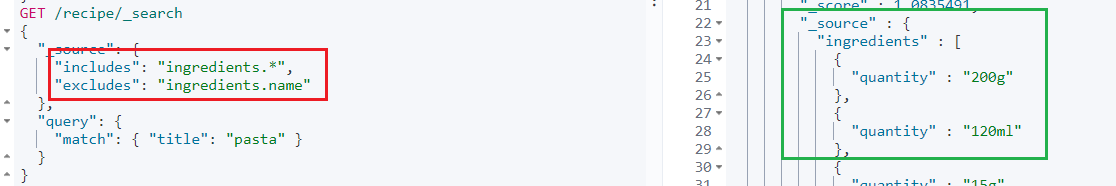
Postoje više načina za kontrolisanje rezultata kod Elasticsearch-a a neki od njih su:

* specificiranje formata; moguće je formatirati rezultat u yaml i json formatu. Formatiranje rezultata u yaml formatu se realizuje u request UI naredbi navođenjem naredbe „format=yaml“ koji je prikazan na prvom primeru na slici 39. S druge strane formatiranje rezultata u json formatu je podrazumevano kod Elasticsearch-a, ali da bi json bio čitljiviji dodaje se naredba „pretty“ u request UI naredbi koji je prikazan u drugom primeru na slici 39:



**Slika 39**: Formatiranje rezultata u yaml i json formatu

* filtriranje „\_source” atributa u rezultatu. Ukoliko upit postane prevelik moguće je pod „\_source“ atributom definisati polja koja je potrebno prikazati u rezultatu. Isto tako moguće je isključititi sva polja iz rezultata prosleđivanjem vrednosti „false“ za atribut „\_source“. Ova naredba se navodi ispred atributa „query“ i ukoliko je naveden u rezultatu se prikazuju samo prosleđena polja sa svih dokumenata. Moguće je unutar „\_source“ atributa i eksplicitno uključiti i isključiti neka polja istovremeno navođenjem atributa „includes“ za uključivanje i „excludes“ za isključivanje polja iz rezultata. Na slici 40 prikazan primer uključivanja svih polja iz niza „ingredients“ osim polja „name“ (označeno crvenom bojom na slici) kao i prikaz rezultata (označeno zelenom bojom na slici):



**Slika 40**: Uključivanje i isključivanje polja u rezultatu

* kontrolisanje maksimalnog broja dokumenata u rezultatu. Ovo je moguće uraditi na dva načina: prvi jeste navođenjem u „size“ u request UI upitu a drugi navođenjem „size“ atributa unutar upita. Na slici 41 prikazana su oba načina sa veličinom 5 (inicijalna veličina za prikaz rezultata je 10 dokumenta kod Elasticsearch-a):



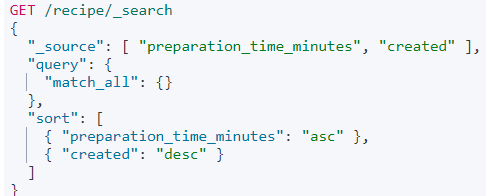
**Slika 41**: Definisanje maksimalnog broja dokumenata u rezultatu

* prikaz sledeće stranice rezultata; Ukoliko je u rezultatu prikazano prvih 5 dokumenta („size“: 5 sa slike 41) moguće je prikazati sledeću stranicu rezultata tj. sledeća 5 dokumenta pomoću atributa „from“. Atribut „from“ u kombinaciji sa „size“ atributom u pozadini kreira prozor sa rezultatima gde „size“ označava koliko se dokumenata prikazuje po stranici dok „from“ označava koja je stranica u pitanju. Na slici 42 prikazan je upit koji prikazuje 5 sledećih dokumenata (na drugoj stranici):



**Slika 42**: prikaz stranice rezultata

* sortiranje rezultata [10]; Reultat je moguće sortirati prema jednom ili više polja navođenjem atribude „sort“. Na slici 43 prikazan je primer sortiranja u rastućem smeru prema polju „preparation\_time\_minutes“ i u opadajućem smeru prema polju „created“:

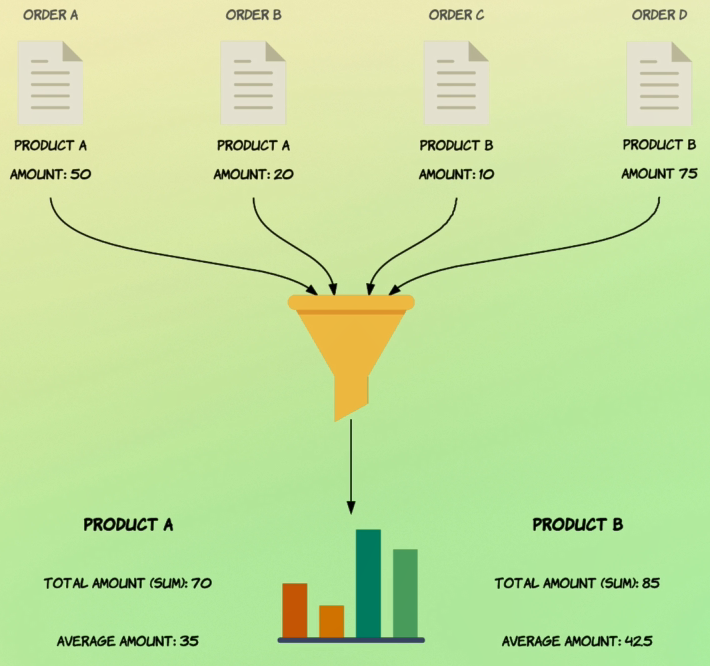


**Slika 43**: Sortiranje rezultata

* filtriranje rezultata pomoću „filter“ atributa;

# Agregacija podataka

Agregacija podataka je znatno moćniji alat kod Elasticsearch-a. Kao što je već poznato agregacija predstavlja grupisanje podataka i izvlačenje statistika iz podataka. Agregacija kod Elasticsearch-a se primenjuje nad skupom dokumenata. Na primer, ukoliko postoji indeks u bazi gde se svaki dokument odnosni na porudzbinu (eng. „Order“) i svaka porudzbina sadrži količinu (eng. „Amount“) i identifikacioni broj proizvoda (eng. „ID“) kao što je prikazano na slici 44:



**Slika 44**: Primer agregacije kod Elasticsearch-a

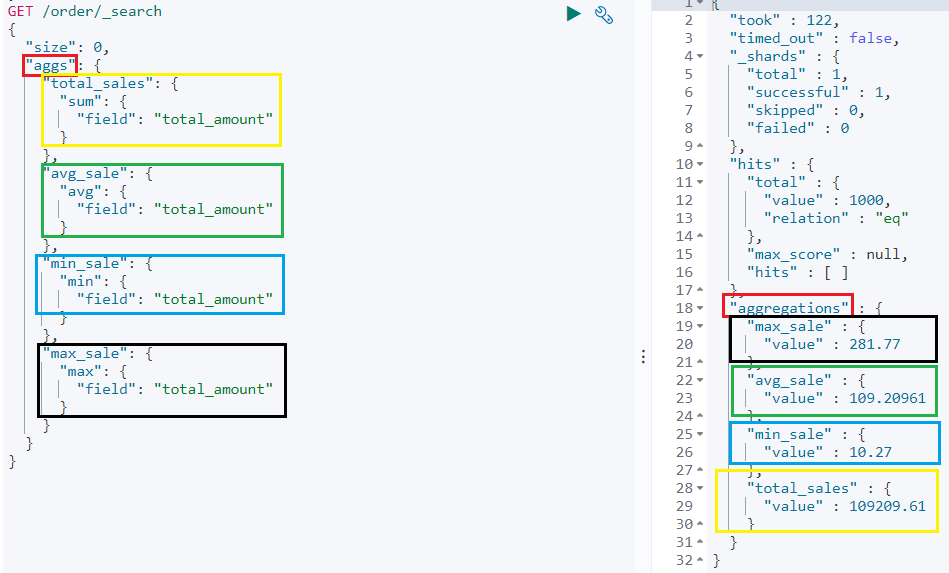
Ovde je moguće primeniti agregaciju grupisanjem porudzbina prema identifikacionom broju proizvoda i da se zatim izvrši sumiranje količine prodatih proizvoda prema njihovom identifikacionom broju kako bi se videlo u kojoj količini je svaki proizvod prodat. Pored sumiranja, izračunati i prosečnu vrednost svakog proizvoda kako bi se videlo koji proizvod najviše donosi profit, kao i niz drugih kompleksih agregacija.

## Metrička agregacija

Metrička agregacija se deli u dve grupe:

1. single-value numeric metric aggregations
2. multi-value numeric metric aggregations

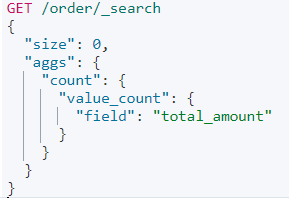
Single-value numeric metric agregacija kao rezultat vraća jednu vrednost koja može da pretdstavlja sumu vrednosti ili prosečnu vrednost itd. S druge strane multi-value numeric metric agregacija vraća više od jedne vrednosti kao rezultat. Agregacija kod Elasticsearch-a se specificira tagom „aggs“ i nekom operacijom na primer „sum“ (sumiranje vrednosti ), „avg“ (prosečna vrednost), „min“ (minimalna vrednost) i „max“ (maksimalna vrednost). Na slici 45 dat je primer agregacije sa navedenim operacijama kao i rezultat agregacije:



**Slika 45**: Primer single-value numeric metric agregacije

Na slici 45, crvenom bojom sa leve strane tagom „aggs“ je označen početak bloka za agregaciju podataka dok je sa desne strane crvenom bojom tagom „aggregations“ označen početak rezultata agregacije. Nad poljem „total\_amount“ se primenjuju sve funkcije za agregaciju. Žutom bojom levo na slici 45 je prikazana operacija sumiranja, dok je na slici desno, takođe žutom bojom prikazan rezultat sumiranja. Svaka boja sa leve strane predstavlja primenu agregacione funkcije dok s desne strane na slici 45 istom bojom je prikazan rezultat primenjene agregacione funkcije.

Za primenu „count“ funkcije kod Elasticsearch-a se koristi atribut „value\_count“ iza koga se navodi polje koje se prebrojava. Na slici 46 je data upotreba „count“ funkcije nad poljem „total\_amount“:



**Slika 46**: Primena count funkcije kod Elasticsearch-a

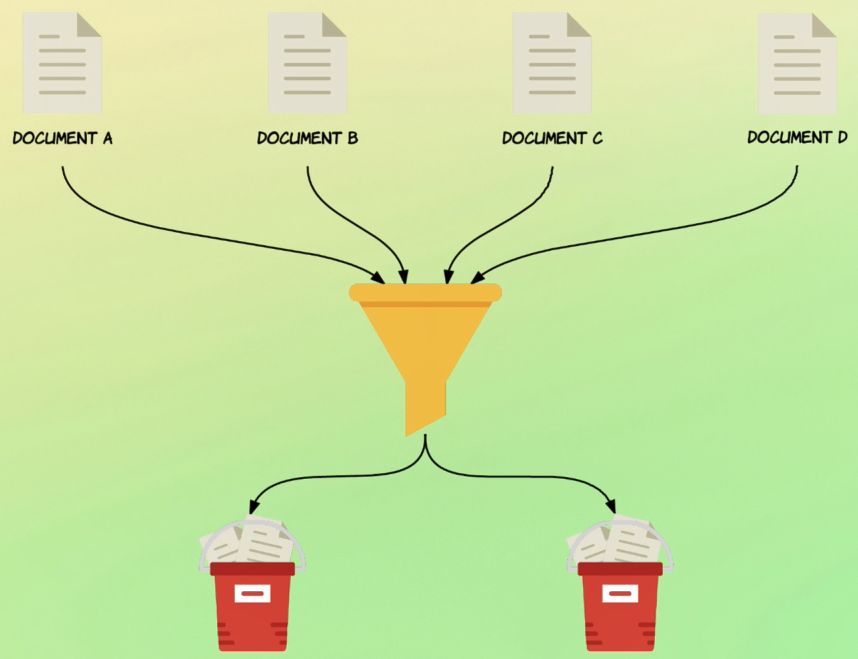
Za multi-value numeric metic agregaciju se koristi atribut „stats” koja vraća rezultat operacija „count“, „min”, „max”, „avg” i „sum” koja je prikazana na slici 47:



**Slika 47**: Primer multi-value numeric metric agregacije

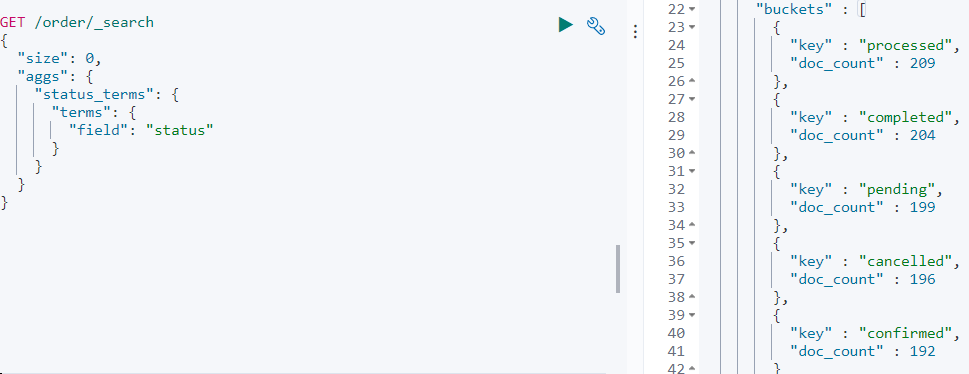
## Bucket agregacija

Bucket agregacija je složenija od metric agregacije. Bucket agregacija funkcioniše tako što gupiše dokumente po kantama (eng. „bucket“). Za svaku kantu unapred definise pravilo koje ukoliko dokument ispunjava smešta u tu kantu. Na slici 48 je prikazan klasičan primer bucket agregacije:



**Slika 48**: Način funkcionisanja bucket agregacije

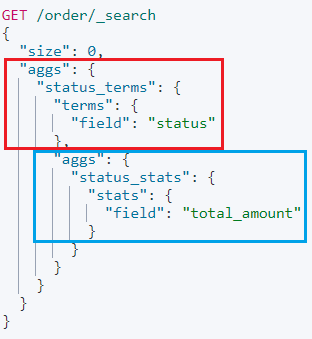
Primer bucket agregacije koja dinamički grupiše dokument u kantama je moguće ostvariti primenom operacije „terms“ kojoj se prosleđuje naziv polja prema kome se vrši grupisanje. Na slici 49 prikazan je primer operacije „terms“ nad poljem „status“:



**Slika 49**: Primer dinamičkog grupisanja dokumenata kod bucket agregacije

## Ugnježdena agregacija

Ugnježdene agregacije je moguće raditi na više načina. U kombinaciji sa bucket agregacijom ugnježdenu agregaciju je moguće primentii kao na prikazanom primeru na slici 50. Na slici 50 se prvo poziva bucket agregacija (označena crvenom bojom) kako bi se grupisali dokumenti po kantama, nakon grupisanja dokumenata se poziva ugnježdena agregacija po kantama nad poljem „total\_amount“ (označena plavom bojom):



**Slika 50**: Primer ugnježdene agregacija sa bucket agregacijom

## Filtriranje agregacije

Kod agregacije moguće je primeniti i filtriranje. Na slici 51, dat je primer izračunavanje prosečne vrednosti „total\_amount“ polja ali da se u prilikom izračunavanja prosečne vrednosti uzmu u obzir samo dokumenti čiji je „total\_amount“ manji od 50 (primenjeno je filtriranje označeno crvenom bojom):



**Slika 51**: Primer filtriranja tokom agregacije podataka

## Ručno definisanje uslova za bucket agregaciju

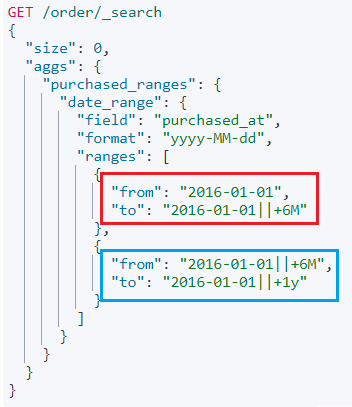
Kod bucket agregacije moguće je definisati niz pravila za agregaciju podataka. Na slici 52, uslov sa skladištenje dokumenata u prvu kantu jeste da document u nazivu sadrži reč „pasta“ a drugi uslov za drugu kantu jeste da u nazivu sadrži reč „spaghetti“. Na samom početu upita je specificirano da je u pitanju agregacija podataka (naveden je atribut „aggs“) nakon čega su primenjeni filteri nad poljem „title“ (označeno crvenom bojom) jedan za ključnu reč „pasta“ a drugi za reč „spaghetti“:



**Slika 52**: Ručno filtriranje dokumenata kod bucket agregacije

## Opseg agregacija

Opseg (eng. „range“) agregacija je veoma slična range upitima. Ovim tipom agregacije moguće je definisati više opsega za koje je potrebno odraditi agregaciju. Na primer moguće je raditi range agregaciju za različite vremenske intervale. Na slici 53, dat je primer za opseg agregaciju dokumenata kupljenih između 01.01. - 01.06.2016. godine (označeno crvenom bojom) i kupljenih između 01.06.2016. - 01.01.2017. godine (označeno plavom bojom):



**Slika 53**: Primer opseg agregacije

## Histogrami

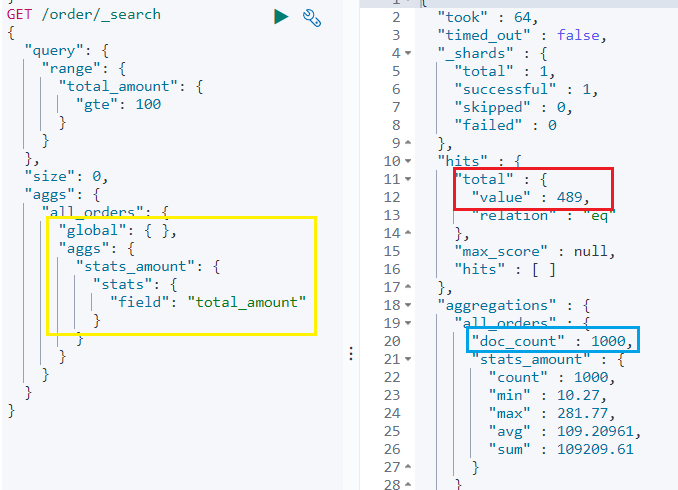
Histogrami dinamički kreiraju kante dokumenata pomoću numeričkih polja u definisanom intervalu. Na primer, za vrednost polja „total\_amount“ moguće je definisati interval 25 na sonovu koga će Elasticsearch kreirati kante dokumenata za svaki interval počev od minimalne vrednotis pa sve do maksimalne vrednosti u polju „total\_amount“. Na slici 54, sa leve strane je kreiran hisogram upit sa intervalom 25 nad poljem „total\_amount“, dok je sa desne strane prikazan rezultat histogram agregacije. U rezultatu je moguće uočiti dva atributa: „key“ koji označava vrednost intervala i atribut „doc\_count“ koji označava broj dokumenata u to intervalu:



Slika 54: Primer histogram agregacije

## Globalna agregacija

Globalna agregacija je veoma korisna kod filtriranja dokumenata. Ključna reč za definisanje globalne agregacije je „global” (označeno žutom bojojm na slici 55). Ukoliko se upitom filtriraju dokumenti čiji je „total\_amount” veći ili jednak od 100 kao na primeru sa slike 55, dodavanjem globalne agregacije moguće je izvući podatak koliko je filtrirano dokumenata (označeno crvenom bojojm) u odnosu na ukupan broj dokumenata (ounačeno plavom bojom):



**Slika 55**: Primer globalne agregacije

## Agregacija ugnježdenih objekata

Navođenjen atributa „nested” moguće je izvršiti agregaciju i nad ugnježdenim objektima u dokumentima. Na slici 56 prikazan je primer agregacije ugnježdenog objekta „employees“. U „department“ dokumentu objekat „employees“ je ugnježden i sadrži više informacija o konkretnom zaposlenom na tom departmanu. Agregacijom je moguće prikazati najmlađeg zaposlenog po departmanu:



**Slika 56**: Agregacija ugnježdenih objekata

# Zaključak

U radu je predstavljen i bliže okarakterisan način pretraživanja podataka i pisanje samih upita kod Elasticsearch-a. Kroz poglavlja su opisani svi načini pretraživanja i dobre prakse za pisanje upita kod Elasticsearch-a. U ovom radu za praktični primer korišćena je Elasticsearch baza podataka i svi upiti su pisani preko Kibana konzolne aplikacije, kako bi upiti i rezultati upita bili jasniji.

Kod Elasticsearch-a je moguće pisati upite na najrazličitije načine za pretraživanje skoro svih tipova podataka, struktuiranih i nestrukturiranih podataka. Automatsko predlaganje upita dodatno olakšava pisanje istih u Kibana konzoli. Upite je moguće primenjivati na skoro svakoj mašini pa i na klasteru sa stotinu čvorova. jer ga krasi velika skalanbilnost pa su performance skoro pa identične na svakoj mašini. Veoma je brz jer koristi distribuirane čvorove i zbog toga pronalazi veoma brzo najbolja poklapanja kako za term level tako i za full text pretraživanje.

# Literatura

1. Apache Lucene veb sajt <https://lucene.apache.org/>
2. Elastic veb sajt <https://www.elastic.co/>
3. Kibana veb sajt <https://www.elastic.co/kibana>
4. Beats veb sajt <https://www.elastic.co/beats/>
5. Logstash veb sajt <https://www.elastic.co/logstash>
6. Elasticsearch veb sajt <https://www.elastic.co/what-is/elasticsearch>
7. Primeri term level upita sa relativnim datumom: [https://www.elastic.co/guide/en/elasticsearch/date\_math](https://www.elastic.co/guide/en/elasticsearch/reference/current/common-options.html#date-math)
8. Regularni izrazi kod Elasticsearch [https://www.elastic.co/guide/en/elasticsearch/query-dsl-regexp-query.html#regexp-syntax](https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-regexp-query.html#regexp-syntax)
9. Full text pretraživanje više polja kod Elasticsearch [https://www.elastic.co/guide/en/elasticsearch/query-dsl-multi-match-query](https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-multi-match-query.html#multi-match-types)
10. Sortiranje rezultata [https://www.elastic.co/sort-search-results](https://www.elastic.co/guide/en/elasticsearch/reference/current/sort-search-results.html)
11. Sortiranje dece [https://www.elastic.co/guide/en/elasticsearch/query-dsl-has-parent-query](https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-has-parent-query.html#_sorting_2)
12. Sortiranje roditelja [https://www.elastic.co/query-dsl-has-child-query](https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-has-child-query.html#_sorting)