Napredne arhitekture informacionih sistema

Pogoni za pretraživanje

Predmetni nastavnik:
dr Marko Vještica

DAIS
Data Science and Information Systems Group

Sadržaj

Pogon za pretraživanje

- Različiti tipovi baza podataka mogu da skladište **velike količine podataka**
  - Ali često **nisu pogodne za punu pretragu teksta** za koju je neophodno:
    - **Indeksiranje** tekstualnog sadržaja
    - **Brza pretraga** teksta koja se **ne odnosi samo na tačna poklapanja** vrednosti zadatih u upitu
    - **Analitičke funkcionalnosti**
- **Pogoni za pretraživanje** (engl. *Search Engine*), poput *Elasticsearch*, projektovani su i optimizovani na način da **upravljaju velikom količinom podataka i omoguće punu pretragu teksta**

Pogon za pretraživanje

- Puna pretraga teksta (engl. *Full-Text Search*) omogućava **obradu upita napisanih na prirodnom jeziku** i vrši se nad **nestrukturiranim tekstualnim sadržajem**
- **Upiti napisani na prirodnom jeziku** od strane korisnika neretko mogu biti **nepotpuni, imati slovnih ili gramatičkih grešaka, sadržati sinonime, biti napisani u drugačijem redosledu** u odnosu na sadržaj u bazi podataka
  - Npr. u relacionim bazama podataka, u slučaju da korisnik napravi slovnu grešku u zadatom stringu, rezultat upita će biti pogrešan ili prazan

Pogon za pretraživanje

- Za razliku od relacione baze podataka, **pogoni za pretraživanje:**
  - Mogu da upravljaju **tekstualnim sadržajem napisanim na različitim jezicima i pismima**
  - **Vrše analizu i procesiranje teksta** kako bi ekstrahovali **semantiku sadržaja**
  - Optimizovani su za pretragu **velike količine nestrukturiranog teksta i upravljanje velikog broja upita**
- Različite baze podataka mogu imati mogućnost kreiranja **indeksa za punu pretragu teksta**
  - Međutim, očekivano je da **pogoni za pretraživanje pruže bolje performanse, skalabilnost i funkcionalnost** u tom domenu jer su optimizovani u svrhu pune pretrage teksta
- Često se **kombinuje relaciona baza podataka** za potrebe transakcione obrade podataka i **pogon za pretraživanje**

Karakteristike i funkcionalnosti pogona za pretraživanje

- Poželjne **karakteristike i funkcionalnosti** koje bi jedan **pogon za pretraživanje** trebalo da ima obuhvataju:
  - **Punu pretragu teksta**
  - Pretragu **geoprostornih** podataka
  - **Brz upis, pretragu i čitanje** podataka
  - **Neosetljivost na slovne greške** u korisničkom upitu
  - **Automatizovano predlaganje reči** (engl. *Autocomplete*) prilikom pretrage
  - **Automatizovana ispravka reči** (engl. *Autocorrection*) prilikom pretrage
  - Jednostavno **horizontalno i vertikalno skaliranje**
  - Visoku **dostupnost sistema**, odnosno otpornost na otkaze
  - Podršku za **primenu različitih modela mašinskog učenja**

Sadržaj

Analiziranje teksta

- Za primenu pune pretrage teksta, potrebno je najpre **procesirati i skladištiti tekstualni sadržaj, odnosno izvršiti:**
  - **Analiziranje** tekstualnog sadržaja
  - **Indeksiranje** tekstualnog sadržaja
- **Analiziranje teksta** predstavlja postupak **dekomponovanja i transformisanja** nestrukturiranog tekstualnog sadržaja koji vrši **analizator teksta**
  - Bez primene analizatora teksta, sadržaj bi bio sačuvan u **originalnom obliku**
    - Time bi se mogućnost pretrage ograničila isključivo na **pronalaženje identičnih poklapanja teksta** (engl. *Exact Match*)
  - Ne bi bilo moguće pronaći **sinonime, reči sa slovnim greškama ili akronime**

Analiziranje teksta

- **Analizator** (engl. *Analyzer*) teksta predstavlja softversku komponentu koja ima **dve osnovne funkcije:**
  - **Tokenizaciju** tekstualnog sadržaja
  - **Normalizaciju** tekstualnog sadržaja
- Obezbeđuje **preprocesiranje teksta** u takvom formatu da je nakon skladištenja moguće pretraživati i **sinonime, akronime, reči koje imaju isti koren, reči sa slovnim greškama**
  - Postoje **različiti tipovi analizatora** koje je moguće koristiti

Tokenizacija

- **Tokenizacija** (engl. *Tokenization*) je proces **dekomponovanja teksta u pojedinačne reči** prateći zadata **pravila dekomponovanja**
  - Npr. podela reči na osnovu **graničnika** poput razmaka
- **Token** (engl. *Token*) predstavlja **pojedinačnu reč** dobijenu postupkom tokenizacije
- **Tokenizator** (engl. *Tokenizer*) predstavlja **softversku komponentu** koja vrši proces tokenizacije
  - Npr. *Standard Tokenizer* dekomponuje rečenice na osnovu razmaka, taba, novog reda ili znakova interpunkcije, koje zatim uklanja

Elasticsearch je pogon za pretragu i nerelaciona, (NoSQL) baza podataka.

[Elasticsearch; je; pogon; za; pretragu; i; nerelaciona; NoSQL; baza; podataka]

Tokenizacija

- **Postoje različiti tokenizatori**, poput:
  - *Whitespace Tokenizer* – dekomponuje rečenice na osnovu razmaka, taba ili novog reda, ali zadržava znakove interpunkcije
  - *N-Gram Tokenizer* – vraća n-grame reči koji se preklapaju
    - Npr. servis -> [se, er, rv, vi, is]
  - *Edge N-Gram Tokenizer* – vraća n-grame reči koji su spojeni sa početkom reči
    - Npr. vektor -> [v, ve, vek, vekt, vekto, vektor]
    - Pogodno za funkcionalnost *autocomplete*
  - ...

Normalizacija

- **Normalizacija** (engl. *Normalization*) je proces **obrade, transformisanja, izmene i obogaćivanja tokena** u cilju postizanja boljeg preklapanja sa različitim upitima
  - Omogućava sistemu da prepozna **širi spektar varijacija u korisničkim upitima**
- Moguće je primeniti **različite tehnike** poput:
  - Prevođenja teksta u **mala slova** (engl. *Lowercase*) – NoSQL -> nosql
  - Uklanjanja **zaustavnih reči** poput rečca i veznika (engl. *Stop Words*) – npr. i, da, kao, pa, ali
    - Uklanjanje reči koje se često pojavljuju u tekstu, a nemaju veliko semantičko značenje
  - Svođenja na **koren reči** (engl. *Stemming*) – npr. pretraga -> trag
    - Omogućava pretragu istih reči navedenih u različitim oblicima
  - Objedinjavanja **sinonima** (engl. *Synonims*) **i akronima** (engl. *Acronyms*) – npr. pogon i motor; SQL i Structured Query Language
  - Svođenja na **ACSII** (engl. *ASCII Folding*) – konverzija *Unicode* znakova koji nisu u prvih 127 *ASCII* znakova (*Basic Latin Unicode Block*) u njihove najbliže *ASCII* ekvivalente – npr. učešće -> ucesce

Normalizacija

- **Svođenje na koren reči** (engl. *Stemming*) predstavlja postupak **uklanjanja prefiksa i sufiksa reči** pomoću unapred **definisanih pravila**, kako bi bile svedene na svoju osnovu, odnosno **koren reči**
  - Jednostavan i brz postupak, ali manje precizan jer **ne posmatra kontekst reči**, a takođe različite porodice reči mogu imati isti koren
- **Svođenje na lemu** (engl. *Lemmatization*) predstavlja postupak **svođenja reči na njen rečnički (izvorni) oblik** (engl. *Dictionary Form*) koji se naziva **lema**
  - Imenice -> nominativ jednine; glagoli -> infinitiv; … (npr. najbolji -> dobar)
  - Precizniji postupak, jer dodatno **posmatra kontekst reči**, ali kompleksniji i sporiji

[Elasticsearch; je; pogon; za; pretragu; i; nerelaciona; NoSQL; baza; podataka]

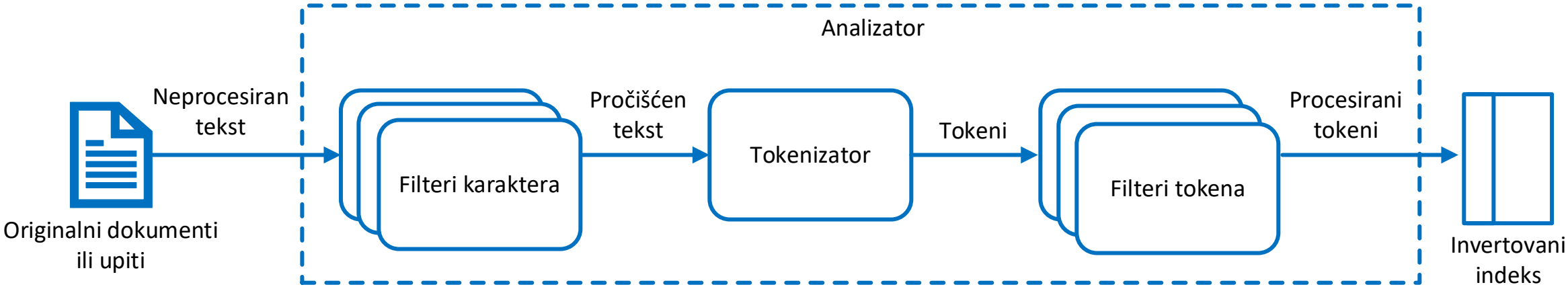
[elasticsearch; gon; trag; relacija; nosql; baz; dat]

Analizator teksta

- **Filteri** (engl. *Filters*) predstavljaju **softverske komponente** koje vrše proces normalizacije – različiti filteri podržavaju **različite tehnike normalizacije teksta**
- **Analizator** teksta sastoji se od **skupa filtera i tokenizatora**, podeljenih u **tri osnovne softverske komponente:**
  - **Filteri karaktera** – **modifikuju** neprocesiran tekst na **nivou karaktera**
    - **Opciona** komponenta, **nula ili više** filtera karaktera
  - **Tokenizator** – **dele** rečenice pročišćenog teksta na **pojedinačne tokene**
    - **Obavezna** komponenta, **tačno jedan** tokenizator
  - **Filteri tokena** – **menjaju i obogaćuju tokene** dobijene od tokenizatora
    - **Opciona** komponenta, **nula ili više** filtera tokena
- **Napomena:** analizator sprovodi procese tokenizacije i normalizacije, ali deo postupka normalizacije sprovodi se pre tokenizacije, a deo nakon tokenizacije

Analizator teksta

- **Svaki dokument ili deo teksta,** prolazi proces analize teksta
  - Vrši se **čišćenje teksta, tokenizacija i izmena tokena**
- **Isti proces prolaze i upiti** koje korisnik postavi
  - Najčešće se **isti analizator** koristi prilikom procesiranja dokumenata i upita

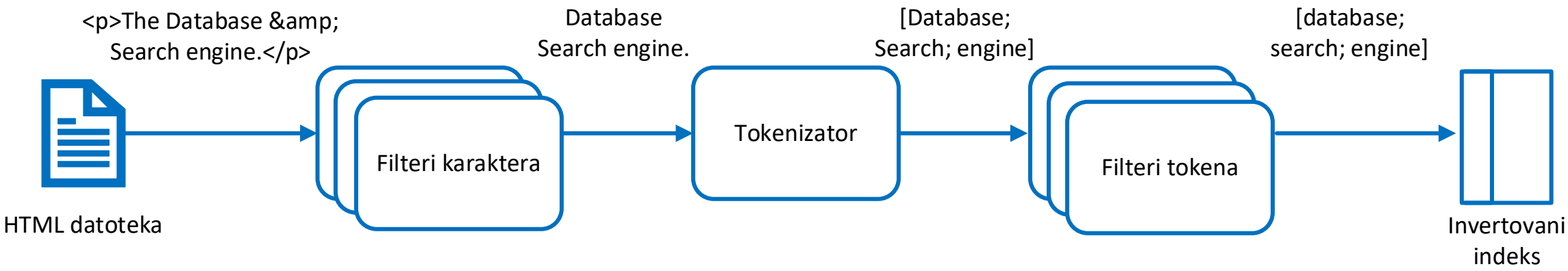


Analizator teksta

- **Filteri karaktera procesiraju karaktere teksta** i mogu da:
  - **Uklanjaju neželjene karaktere** neprocesiranog teksta (npr. HTML tagove)
  - **Zamene deo teksta** drugim tekstom (npr. & -> and; α -> alpha)
  - Na osnovu šablona zamene deo teksta pomoću **regularnog izraza** – *regex* (npr. poklapanje e-adrese i ekstrahovanje domena organizacije)
- **Tokenizator deli procesirani tekst na tokene** na osnovu zadatog **graničnika**
  - **Graničnik** može predstavljati razmak, znak interpunkcije, bilo koji karakter koji nije slovo
  - Npr. podrazumevani *standard tokenizer* izdvaja reči na osnovu gramatike i znakova interpunkcije
  - **Tokenima** mogu biti dodeljeni različiti **meta-podaci** koji ih opisuju

Analizator teksta

- **Filteri tokena procesiraju tokene** dobijene od tokenizatora i mogu da:
  - Menjaju velika u mala slova; kreiraju sinonime reči; svode na koren reči; proizvode n-grame; uklanjaju duplikate reči…
- Postoje **različiti filteri i tokenizatori** koje nude pogoni za pretraživanje
  - Mogu takođe biti iskorišćeni filteri i tokenizatori razvijeni nezavisno od pogona



Analizator teksta

- Svaki **token** sadrži **meta-podatke** poput tipa (npr. *alphanum*, *num*, *emoji*); pozicije tokena u tekstu; i rednog broja početnog i krajnjeg karaktera tokena u tekstu
- **Primer tokena** dobijenih od *standard analyzer*

  „Baze podataka 3 💾 !“ -> [baze; podataka; 3; 💾]

token	start_offset	end_offset	type	position
baze	0	4	ALPHANUM	0
podataka	5	13	ALPHANUM	1
3	14	15	NUM	2
💾	16	17	EMOJI	3

- Moguće je uz pomoć drugih analizatora ili ručnih podešavanja **preslikati emodži na reči** (npr. 💾 -> skladištenje podataka)

Indeksiranje tokena

- **Nakon procesa analiziranja teksta**, odnosno transformisanja neprocesiranog teksta u tokene, sledi **postupak indeksiranja tokena**
  - Dobijeni tokeni **evidentiraju se u invertovanim indeksima** baze podataka pogona za pretraživanje radi postizanja **dobrih performansi pune pretrage teksta**
- **Invertovani indeks** preslikava **tokene na dokumente** u kojima se nalaze
  - Poput indeksa reči na kraju knjige
  - Predstavlja **glavnu komponentu** za potrebe **pune pretrage teksta**
    - Time se obezbeđuje brza pretraga teksta jer se **ne pristupa sadržaju direktno**, nego posredstvom invertovanog indeksa
    - Moguće je takođe **distribuirati indeks na različite servere** i time dodatno ubrzati pretragu
- **Napomena:** postoje i druge strukture podataka koje se koriste za različite pretrage, poput BKD stabla (engl. *Block k-dimensional trees*) koje se koriste za numeričke i geoprostorne podatke

Invertovani indeks

- **Invertovani indeks** (engl. *Inverted Index*) predstavlja **strukturu podataka** koja:
  - Sadrži **listu svih jedinstvenih tokena** koji se **pojavljuju u dokumentima**
  - A svaki **token pokazuje na listu dokumenata** u kojima se pojavljuje kao i njegovu **učestalost pojavljivanja u dokumentima**
- Postoji invertovani indeks za **svako polje tipa tekst** u bazi podataka pogona
  - Može se posmatrati kao **mapa**, u kojoj **ključ predstavljaju tokeni, a vrednost listu dokumenata**
- Najčešće se **ne modifikuje** prilikom ažuriranja dokumenata, jer zahteva **velike izmene koje su vremenski zahtevne**
  - Najčešće je **nepromenljiv** i koristi se samo za **čitanje**
  - Ukoliko postoji potreba za ažuriranjem indeksa, potrebno ga je **ponovo kreirati**
  - Moguće je **periodično ažuriranje indeksa** od strane pogona za pretraživanje
- **Terminom** (engl. *Term*) se naziva **token** koji je evidentiran u **invertovanom indeksu**

Doc. ID 0

Graph database is NoSQL database that stores node and relation data

Doc. ID 1

Vector database stores vector data

Doc. ID 2

Data node stores data and searches data

Identifikator termina	Termin	Identifikatori dokumenata u kojima se termin nalazi i broj njegovog pojavljivanja	Broj dokumenata koji sadrže termin
1	and	[0, 1], [2, 1]	2
2	data	[0, 1], [1, 1], [2, 3]	3
3	database	[0, 2], [1, 1]	2
4	graph	[0, 1]	1
5	is	[0, 1]	1
6	node	[0, 1], [2, 1]	2
7	nosql	[0, 1]	1
8	relation	[0, 1]	1
9	searches	[2, 1]	1
10	stores	[0, 1], [1, 1], [2, 1]	3
11	that	[0, 1]	1
12	vector	[1, 2]	1

Invertovani indeks

- U prethodnom **primeru:**
  - Ukoliko tražimo reči „*vector database*“, prva dva dokumenta će biti dobavljena
  - Ali drugi dokument bi trebalo da je relevantniji u odnosu na prvi jer sadrži obe reči
- **Pogoni za pretraživanje** ne vraćaju samo rezultat na osnovu upita, već vraćaju **rezultat koji je uređen po oceni relevantnosti dokumenata**
- **Ocena relevantnosti** (engl. *Relevancy Score*) rezultata pune pretrage teksta predstavlja **pozitivan decimalan broj** na osnovu kojeg se vrši **rangiranje rezultata**
  - **Invertovani indeks** pomaže u određivanju ocene relevantnosti dokumenata jer **sadrži učestalost pojavljivanja termina**, kao i **broj dokumenata u kojima se pojavljuje**

Sadržaj

Algoritmi za ocenu relevantnosti dokumenata

- Pogoni za pretraživanje koriste **različite algoritme za ocenu relevantnosti dokumenata** prilikom postavljanja upita, kao što su:
  - ***Term Frequency – Inverse Document Frequency* (TF-IDF)**
    - Korišćen kao podrazumevani algoritam u pogonu *Elasticsearch* do verzije 5
  - ***Okapi Best Match* 25 (BM25)**
    - Trenutno se koristi kao **podrazumevani algoritam** u mnogim sistemima (npr. *Elasticsearch* od verzije 5)
    - Predstavlja **napredniju varijantu algoritma TF-IDF**
- Postoje i **drugi algoritmi za ocenu relevantnosti dokumenata** koji se koriste
  - Razlikuju se po tome **na koji način određuju težinu termina** prilikom pretrage

Algoritmi za ocenu relevantnosti dokumenata

- **Tri glavna faktora** učestvuju u određivanju ocene relevantnosti dokumenta u oba algoritma:
  - **Učestalost pojavljivanja termina u dokumentu**
  - **Dužina dokumenta**
  - **Relevantnost termina na osnovu njegove zastupljenosti u svim dokumentima**
- **Napomena:** često se navodi dokument kao jedinica koja se pretražuje u pogonima za pretraživanje koji podatke čuvaju u formi JSON, a zapravo se vrši **pretraga na nivou polja**

TF-IDF

- **TF-IDF** predstavlja algoritam za određivanje ocene relevantnosti dokumenata
- Sadrži **dve komponente** koji utiču na svaki pojedinačni termin:
  - **TF** – **Učestalost termina** (engl. *Term Frequency*)
    - **Broj pojavljivanja** termina u dokumentu
  - **IDF** – **Inverzna učestalost dokumenta** (engl. *Inverse Document Frequency*)
    - **Mera zastupljenosti** termina u svim dokumentima
- Termin koji se **često pojavljuje u konkretnom dokumentu**, a **retko se pojavljuje u svim ostalim dokumentima** smatra se **više relevantnim**

TF-IDF – učestalost termina

- **Učestalost termina (TF)** – **broj pojavljivanja** termina u dokumentu
  - Što je **učestalost termina veća**, to je **ocena relevantnosti veća**

$$TF(t, d) = freq(t, d)$$

$$t \ - traženi\ termin,$$
$$d \ - dokument\ u\ kojem\ se\ vrši\ pretraga$$

  - Izračunava se najčešće prilikom **indeksiranja dokumenata**, a ne tokom pretrage
    - Skladišti se u **invertovanom indeksu** kako bi izvršavanje upita bilo performantno
- **Problem:** što je dokument obimniji, **reči teže više puta da se pojave**, ali to **ne znači da je obimniji dokument relevantniji od kraćeg dokumenta**
- Npr. u upitu pretrage je zadata reč „vektor“
  - Dokument 1: 10.000 reči, 30 pojavljivanja reči „vektor“, TF(vektor, d1) = 30
  - Dokument 2: 200 reči, 20 pojavljivanja reči „vektor“, TF(vektor, d2) = 20

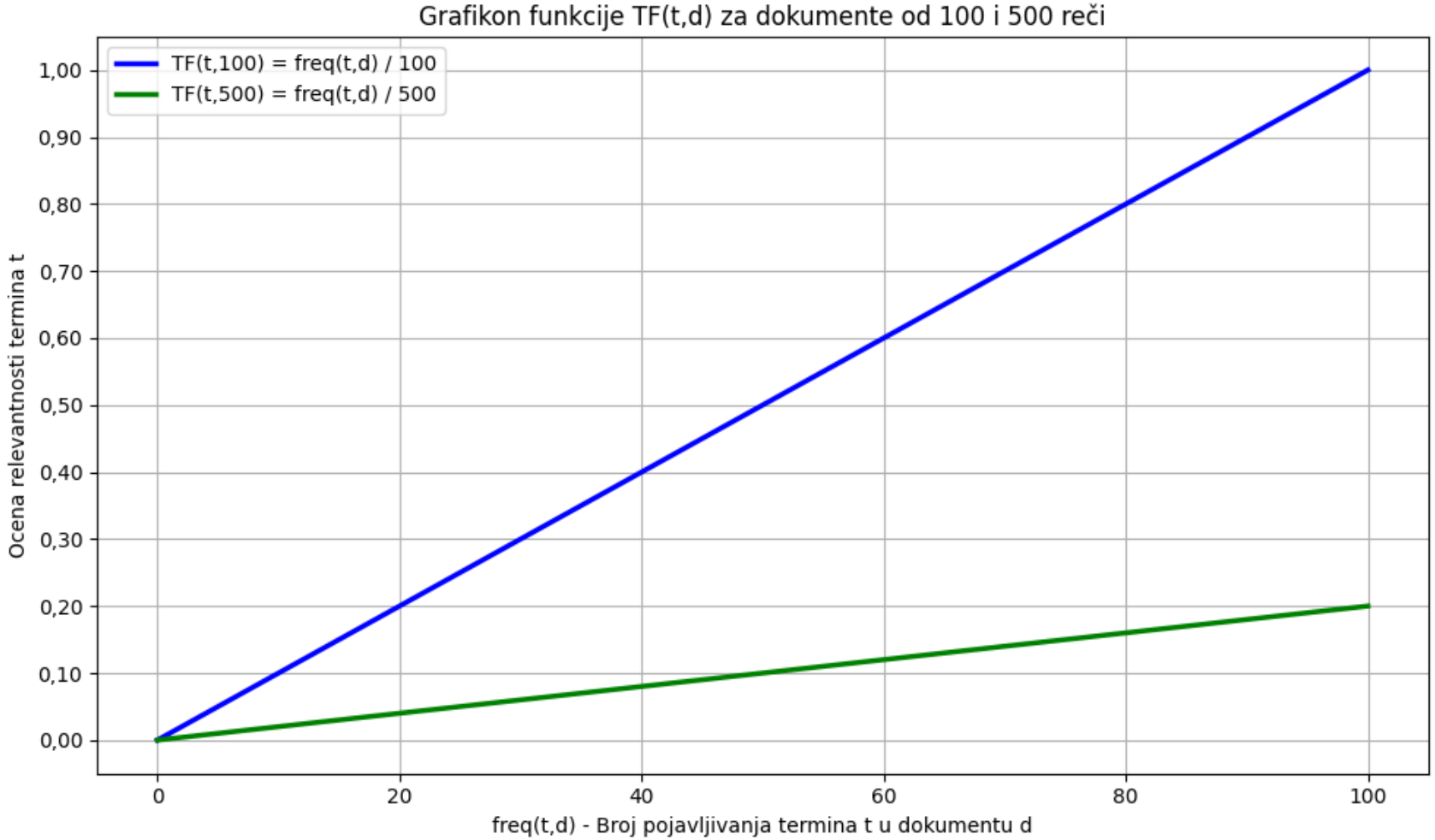
TF-IDF – učestalost termina

- **Funkcija TF normalizuje se dužinom dokumenta** (engl. *Field-Length Norm*) kako bi se **umanjilo davanje većeg značaja obimnijim dokumentima**
  - Normalizacijom TF **smanjuje se relevantnost** učestalosti termina u **obimnim dokumentima**, a **povećava se u manjim dokumentima**
- **Dužina dokumenta** predstavlja **ukupan broj termina** u dokumentu

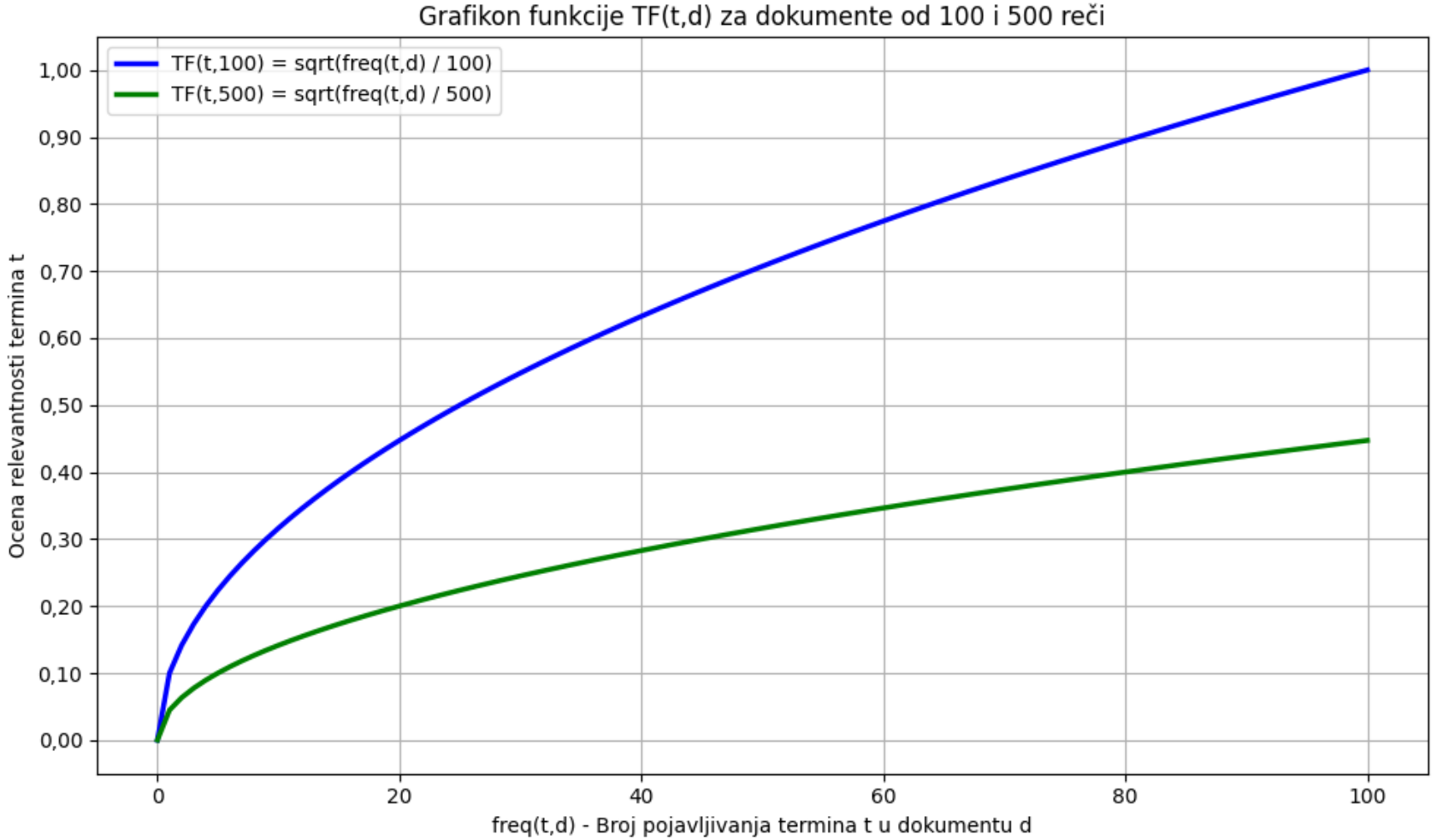
$$TF(t,d) = \frac{freq(t,d)}{count(d)} \qquad count(d) = \sum_{t' \in d} freq(t',d)$$

- Npr. u upitu pretrage je zadata reč „vektor“
  - Dokument 1: 10000 reči, 30 pojavljivanja reči „vektor“, TF(vektor, d1) = 0,003
  - Dokument 2: 200 reči, 20 pojavljivanja reči „vektor“, TF(vektor, d2) = 0,1
- U pojedinim varijantama algoritma, koristi se **kvadratni koren** učestalosti termina
  - Kako bi umanjio značaj visoke učestalosti jednog termina u odnosu na ostale učestalosti
  - Prethodni primer: TF(vektor, d1) = 0,055; TF(vektor, d2) = 0,316 (smanjena razlika između ocena)

TF-IDF – učestalost termina



TF-IDF – učestalost termina



TF-IDF – učestalost dokumenta

- Da li je svaki **termin istog značaja?**
  - Npr. reč „vektor“ i veznik „i“
- Funkcija **TF ne razlikuje značaj termina** u dokumentima
  - Zbog čega se uvodi inverzna učestalost dokumenta (IDF)
- **DF – Učestalost dokumenta** (engl. *Document Frequency*) predstavlja **broj dokumenata** u kojima se traženi **termin pojavljuje**
  - Obuhvataju se svi dokumenti u **kolekciji** koja se pretražuje
    - U pogonu *Elasticsearch* koristi se termin **indeks** za kolekciju dokumenata
  - Što je **vrednost DF veća**, znači da je **termin uobičajen u dokumentima** i da **nema veliki značaj**
  - Npr. na engleskom „the“, „a“, „and“ nisu toliko značajni prilikom pretrage

TF-IDF – inverzna učestalost dokumenta

- **Inverzna učestalost dokumenta (IDF)** – mera **odnosa ukupnog broja dokumenata** i broja dokumenata koji **sadrže traženi termin**
  - Što je **vrednost IDF veća**, znači da se **termin retko pojavljuje** u dokumentima i da je **značajniji**
  - **Povećanjem broja pojavljivanja** termina u različitim dokumentima, čini termin **manje značajnim**

$$IDF(t, D) = \frac{|D|}{num(t, D)} \qquad num(t, D) = |\{d \mid d \in D \wedge t \in d\}|$$

t – *traženi termin*,
D – *skup svih dokumenata*

- **Problem: značajnost termina veoma opada** kako se povećava broj dokumenata u kojima se nalazi
  - Npr. značajnost termina opada duplo ukoliko se umesto u jednom, pojavi u dva dokumenta

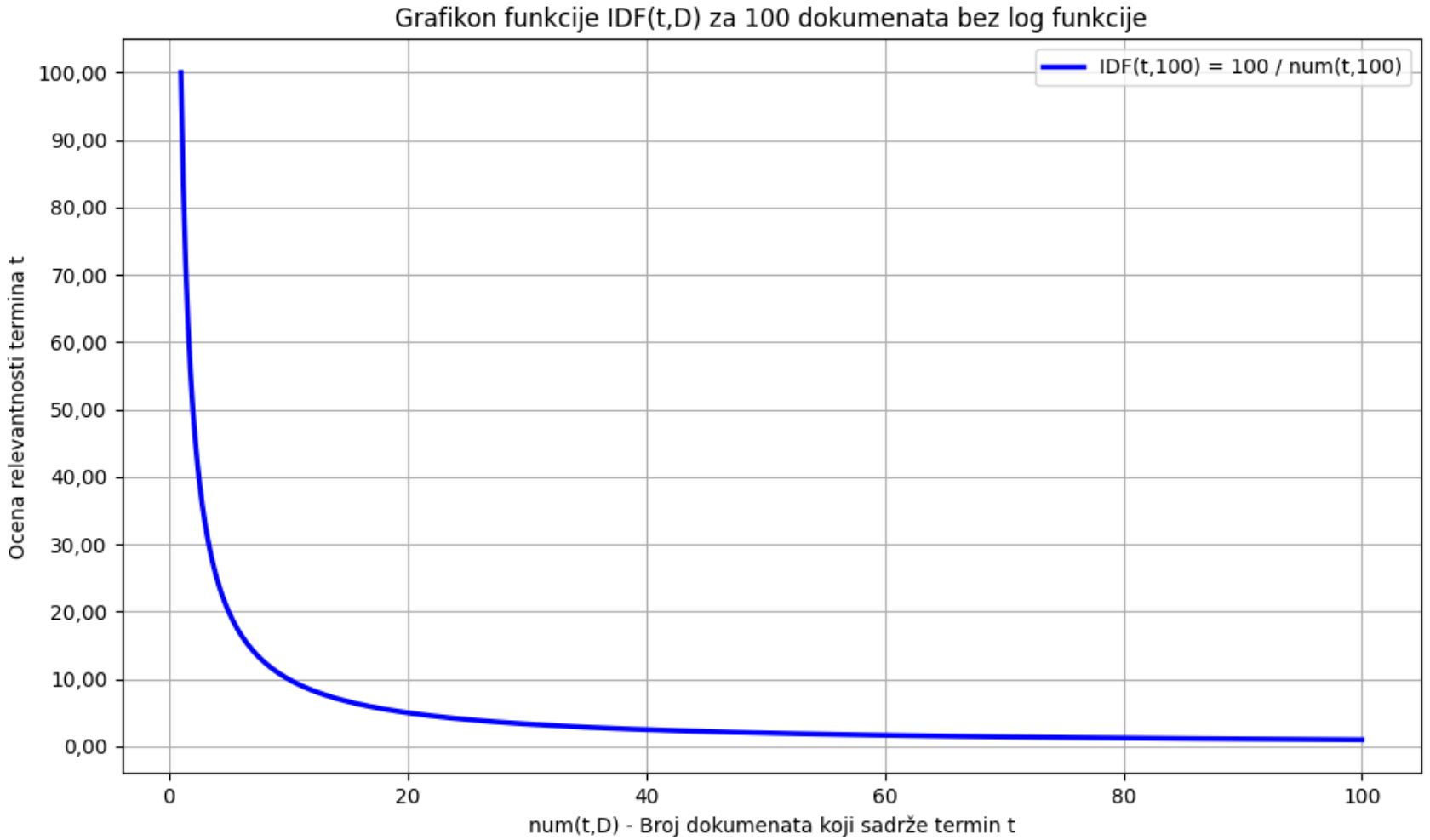
TF-IDF – inverzna učestalost dokumenta

- Uvodi se **logaritamska funkcija** za vrednost IDF koja **smanjuje nagli pad značaja termina** prilikom povećanja broja dokumenata u kojima se pojavljuje

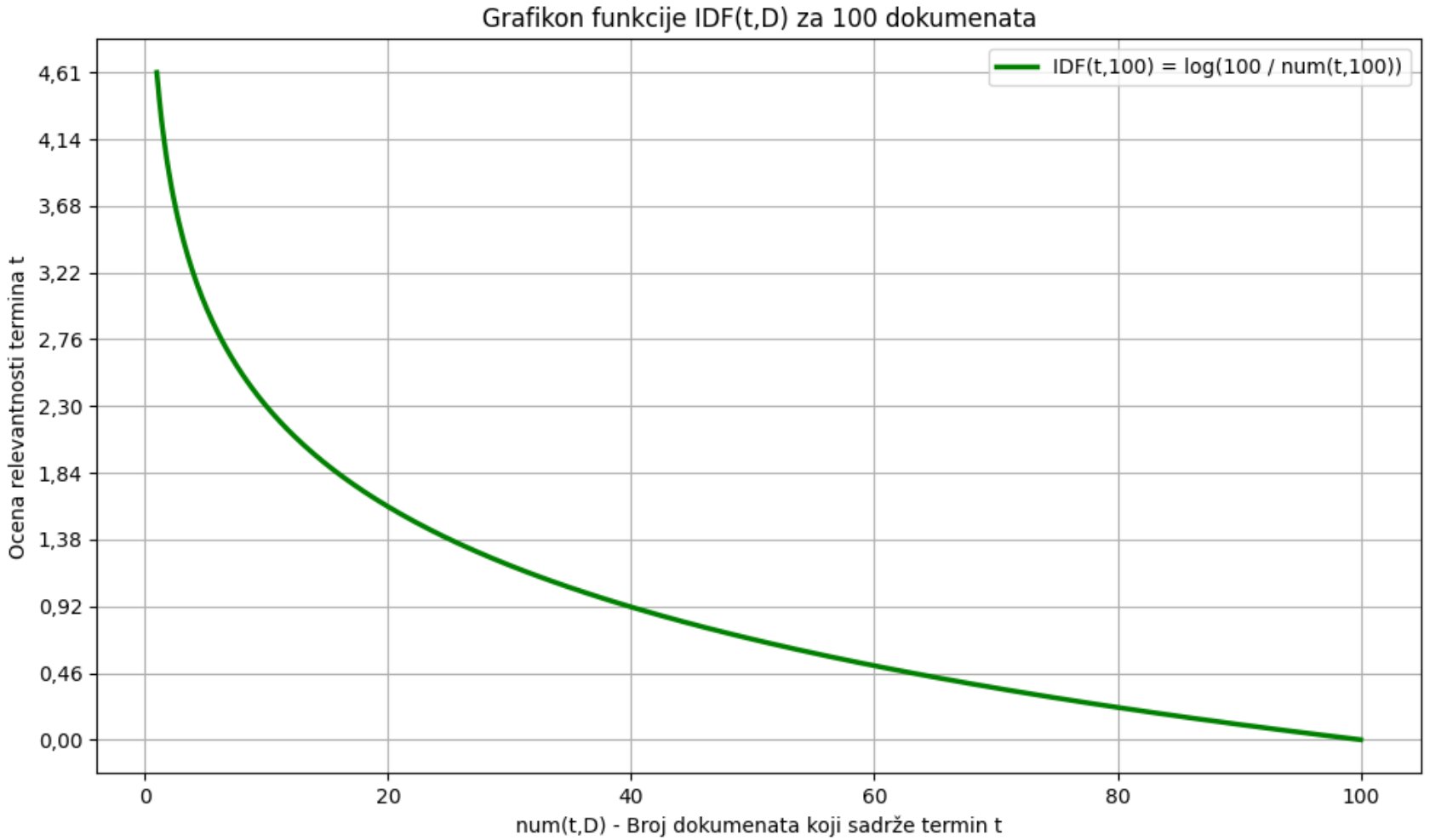
$$IDF(t, D) = \log\left(\frac{|D|}{num(t, D)}\right)$$

- Funkcija IDF računa se na nivou dokumenata **u okviru jednog indeksa**
- Ukoliko se termin pojavi u **svim dokumentima indeksa**, vrednost **IDF je 0**

TF-IDF – inverzna učestalost dokumenta



TF-IDF – inverzna učestalost dokumenta



TF-IDF

- **TF-IDF** predstavlja **proizvod učestalosti termina** u dokumentu i **značaja termina** u korpusu dokumenata

$$TFIDF(t, d, D) = TF(t, d) * IDF(t, D)$$

- Moguće su sledeće **varijante algoritma TF-IDF:**
  - U slučaju da se traženi **termin pojavljuje u svim dokumentima**, tada bi vrednost IDF bila 0, a takođe i celokupna vrednost **TF-IDF bi bila 0**, stoga se dodaje 1 na vrednost IDF
  - Prilikom postavljanja upita, moguće je zadati **termin koji ne postoji ni u jednom dokumentu**, tada bi **učestalost dokumenta** $num(t, D)$ **bila jednaka nuli**, stoga se dodaje 1 na učestalost dokumenta kako bi se sprečilo deljenje sa nulom
    - Druga opcija je da **termin bude ignorisan** prilikom pretrage

$$IDF(t, D) = 1 + \log\left(\frac{|D|}{1 + num(t, D)}\right)$$

TF-IDF – ocena relevantnosti za upit

- **Rezultat TF-IDF** koristi se kao **ocena relevantnosti dokumenata** na osnovu **termina zadatih u upitu**
  - Koristi se za potrebe **rangiranja dokumenata** i vraćanja **najrelevantnijih rezultata**
- **Za svaki dokument** u skupu dokumenata (indeksu), računa se **suma vrednosti TF-IDF** za svaki termin u upitu, nakon čega se vrši **rangiranje dokumenata** na osnovu **sumirane ocene relevantnosti pojedinačnih termina**

$$s(Q,d,D) = \sum_{i=1}^{n} TFIDF(q_i,d,D) = \sum_{i=1}^{n} TF(q_i,d) * IDF(q_i,D)$$

Q – termini postavljenog upita,
q – pojedinačni termin upita,
n – broj termina u upitu,
D – skup svih dokumenata u indeksu,
d – dokument za koji se računa ocena relevantnosti svih termina iz upita

Doc. ID 0

Graph database is NoSQL database that stores node and relation data

$$TFIDF(graph\ database\ data) = \frac{1}{11} * \log\left(\frac{3}{1}\right) + \frac{2}{11} * \log\left(\frac{3}{2}\right) + \frac{1}{11} * \log\left(\frac{3}{3}\right) =$$

$$= 0{,}091 * 0{,}477 + 0{,}182 * 0{,}176 + 0{,}091 * 0 \approx 0{,}075$$

Doc. ID 1

Vector database stores vector data

$$TFIDF(graph\ database\ data) = \frac{0}{5} * \log\left(\frac{3}{1}\right) + \frac{1}{5} * \log\left(\frac{3}{2}\right) + \frac{1}{5} * \log\left(\frac{3}{3}\right) =$$

$$= 0 * 0{,}477 + 0{,}2 * 0{,}176 + 0{,}2 * 0 \approx 0{,}035$$

Doc. ID 2

Data node stores data and searches data

$$TFIDF(graph\ database\ data) = \frac{0}{7} * \log\left(\frac{3}{1}\right) + \frac{0}{7} * \log\left(\frac{3}{2}\right) + \frac{3}{7} * \log\left(\frac{3}{3}\right) =$$

$$= 0 * 0{,}477 + 0 * 0{,}176 + 0{,}429 * 0 = 0$$

BM25

- **BM25 naprednija je varijanta algoritma TF-IDF** za određivanje ocene relevantnosti dokumenata
- Uvodi **dve bitne izmene** u osnovnom TF-IDF algoritmu:
  - **Zasićenje učestalosti termina** – koristi **nelinearnu funkciju za učestalost termina** kako bi termini nakon određenog broja ponavljanja dobili visoku ocenu i ušli u zasićenje
  - **Uticaj dužine dokumenta na učestalost termina** – uvodi **faktor normalizacije dužine dokumenta** kako bi termini kraćih dokumenata brže ušli u zasićenje, a termini dužih dokumenata sporije ušli u zasićenje
- BM25 dodatno **smanjuje uticaj dužih dokumenata**
- Očekivano je da BM25 vrati **preciznije i relevantnije rezultate** u odnosu na TF-IDF
  - Ali je **kompleksniji za izračunavanje**

BM25 – učestalost termina

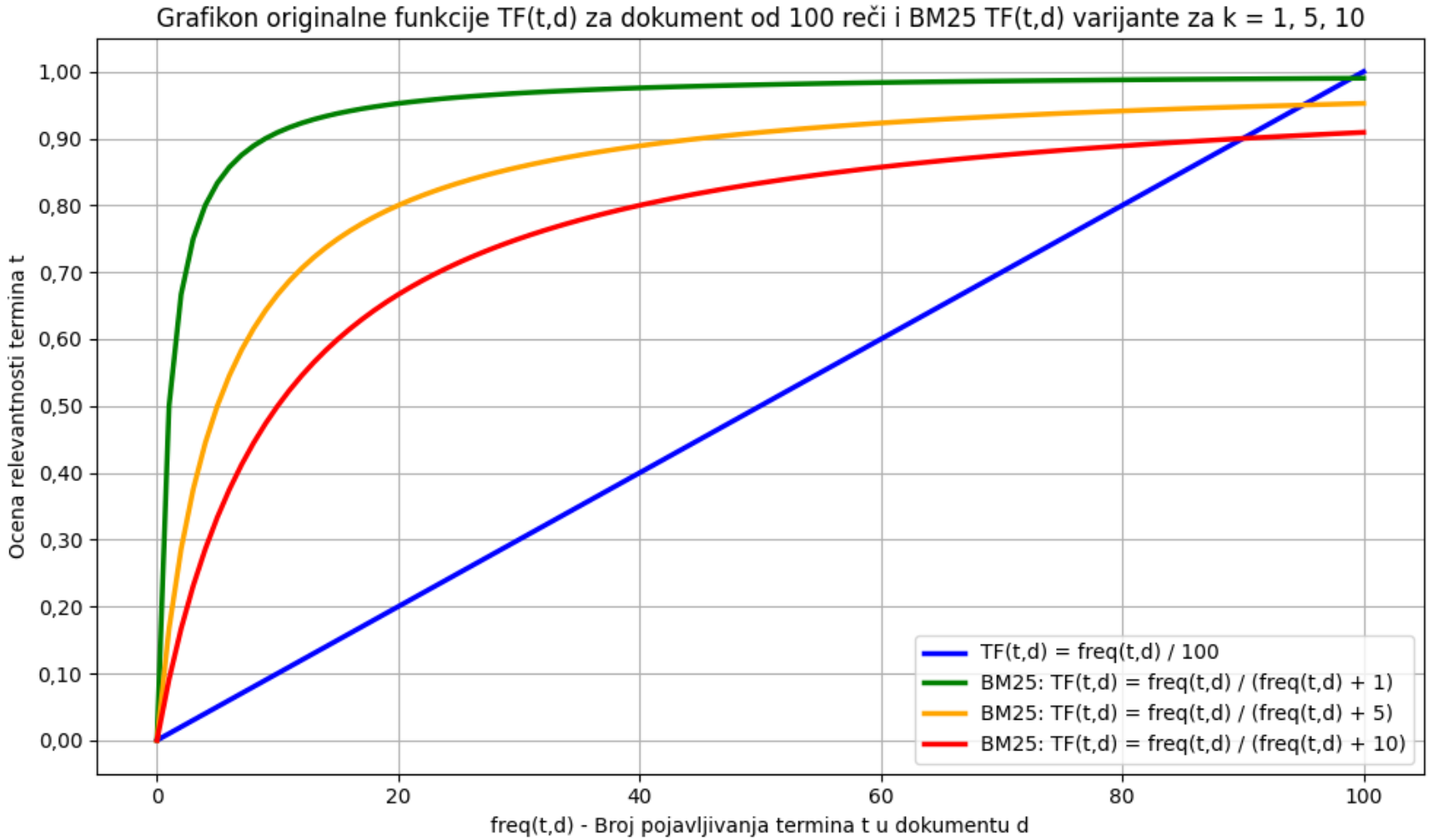
- **Problem linearnog rasta ocene učestalosti termina** algoritma TF-IDF
  - Kako **broj pojavljivanja termina** u dokumentu **raste**, linearno raste i vrednost TF
  - Npr. u jednom dokumentu termin „vektor“ pojavi se 100 puta, a drugom dokumentu 200 puta, da li to znači da je drugi dokument duplo relevantniji?
- **Nije očekivano** da relevantnost termina u dokumentu **linearno raste** u odnosu na njegov **broj pojavljivanja u dokumentu**
  - **Veća razlika u relevantnosti postoji između malih brojeva ponavljanja termina**, nego između velikih brojeva ponavljanja termina
    - Odnosno, kako **broj pojavljivanja** određenog termina u dokumentu **raste**, njegova **relevantnost sporije raste** – dolazi u zasićenje
  - Npr. očekivano je da bude veća razlika u relevantnosti ukoliko se u jednom dokumentu termin pojavio 3 puta, a u drugom 6 puta, nego kada se u jednom dokumentu pojavio 200, a u drugom 400 puta
    - **TF-IDF nije osetljiv** na navedenu razliku u primeru

BM25 – učestalost termina

- Algoritam BM25 uvodi **zasićenje relevantnosti broja pojavljivanja termina** kako bi veći značaj dao razlikama u malom broju pojavljivanja termina
  - Nakon prelaska određenog **praga broja pojavljivanja**, što se termin više pojavljuje u dokumentu, njegov **uticaj na ocenu relevantnosti termina se gotovo ne menja** (skoro ostaje konstantan)
  - Npr. postoji velika razlika uticaja ako se neki termin pojavio 2 ili 4 puta, ali je gotovo identičan uticaj termina ukoliko se pojavio 40 ili 80 puta
- **Zasićenje pojavljivanja termina** postiže se uvođenjem **parametra *k*** koji stvara veći uticaj na manji broj pojavljivanja termina, a manji uticaj na veći broj pojavljivanja termina
  - **Manji** parametar *k* – **brže zasićenje** termina; **veći** parametar *k* – **sporije zasićenje** termina
  - Bira se vrednost **veća od 0** (za vrednost 0, TF ne bi imao uticaj na ocenu relevantnosti, samo IDF)

$$TF_{BM25}(t, d, k) = \frac{freq(t, d)}{freq(t, d) + k}$$

BM25 – učestalost termina



BM25 – učestalost termina

- BM25 uvodi dodatno unapređenje time što u funkciju uključuje i **normalizovanu dužinu dokumenta**
  - Parametar *k* množi se sa **normalizovanom dužinom dokumenta**
    - Predstavlja **odnos dužine dokumenta i prosečne dužine dokumenta**
  - Na taj način:
    - **Termini kraćih dokumenata brže dolaze do zasićenja**, jer u slučaju dokumenata čija je dužina manja od prosečne, parametar *k* množi se sa vrednošću manjom od 1
    - **Termini obimnijih dokumenata sporije dolaze do zasićenja**, jer u slučaju dokumenata čija je dužina veća od prosečne, parametar *k* množi se sa vrednošću većom od 1
  - Npr. veći je uticaj razlike između 2 i 4 broja pojavljivanja termina u dokumentu od 100 termina, nego u dokumentu od 400 termina

$$TF_{BM25}(t,d,k,D) = \frac{freq(t,d)}{freq(t,d) + k * \frac{count(d)}{avg(D)}} \qquad avg(D) = \frac{1}{n}\sum_{i=1}^{n} count(d_i)$$

BM25 – učestalost termina

- Kako nisu svi domeni isti i **ne može se dužina dokumenta posmatrati isto u svim domenima**, BM25 uvodi **parametar *b*** za podešavanje **uticaja dužine dokumenta**
  - U nekim domenima **dužina dokumenta je veoma važna, dok u drugim nije**
  - Ukoliko je **vrednost parametra *b* jednaka:**
    - **0**, tada **dužina dokumenta nema uticaj na zasićenje termina**
    - **1**, tada **dužina dokumenta ima najveći uticaj na zasićenje termina**
  - Što je vrednost **parametra *b* veća**, to je **uticaj dužine dokumenta** na zasićenje termina sve **veći**
    - Moguće je podešavati uticaj dužine dokumenata odabirom vrednosti parametra *b* **između 0 i 1**

$$TF_{BM25}(t,d,k,D,b) = \frac{freq(t,d)}{freq(t,d) + k * \left(1 - b + b * \frac{count(d)}{avg(D)}\right)}$$

BM25 – inverzna učestalost dokumenta

- BM25 algoritam takođe **modifikuje IDF da bolje prepozna retke termine**
  - U formulu se uključuju **konstantne vrednosti,** a od ukupnog broja dokumenata se **oduzima broj dokumenata u kojima se traženi termin nalazi**
  - Modifikacija je nastala **empirijski**, kako ne bi morali da se podešavaju različiti parametri za IDF, već je ostavljena funkcija koja **u većini slučajeva daje dobre rezultate**

$$IDF_{BM25}(t, D) = \log\left(\frac{|D| - num(t, D) + 0{,}5}{num(t, D) + 0{,}5}\right)$$

- **Problem:** funkcija vraća **negativnu vrednost** ako se **termin pronađe u više od polovine dokumenata**, stoga se dodaje **vrednost 1**

$$IDF_{BM25}(t, D) = \log\left(1 + \frac{|D| - num(t, D) + 0{,}5}{num(t, D) + 0{,}5}\right)$$

BM25

- BM25 predstavlja **proizvod modifikovanih TF i IDF komponenti**

$$BM25(t,d,k,D,b) = TF_{BM25}(t,d,k,D,b) * IDF_{BM25}(t,D)$$

$$BM25(t,d,k,D,b) = \frac{freq(t,d)}{freq(t,d) + k * \left(1 - b + b * \frac{count(d)}{avg(D)}\right)} * \log\left(1 + \frac{|D| - num(t,D) + 0{,}5}{num(t,D) + 0{,}5}\right)$$

- Moguće je **podešavati parametre *k* i *b***
  - **Parametar *k* – faktor nelinearnog zasićenja učestalosti termina**
    - Utiče na **brzinu zasićenja termina**
  - **Parametar *b* – faktor normalizacije učestalosti termina zasnovan na dužini dokumenta**
    - Utiče na **značaj dužine dokumenta**

BM25 – podešavanje parametara *k* i *b*

- **Podrazumevane vrednosti** (*k*=1,2 i *b*=0,75) daju **dobre rezultate u većini domena**
  - **Empirijski** zaključeno da je najbolje upotrebiti **parametar *k* u rasponu od 0,5 do 2**, dok je **parametar *b*** najbolje upotrebiti **u rasponu od 0,3 do 0,9**
  - **Neophodno je testiranje rezultata upita** prilikom bilo kakve promene parametara *k* ili *b*
- Kako odabrati **vrednost parametra *k*?** Pitanje je koliko su **dokumenti obimni** – odnosno kada je očekivano da termin dođe u zasićenje
  - Ukoliko su u korpusu **obimni dokumenti** (npr. knjige)
    - Traženi **termini teže više puta da se pojavljuju u dokumentu**, iako pojedini termini možda **nisu toliko relevantni** u konktekstu upita
    - Zadaje se **veća vrednost parametra *k*,** kako **zasićenje ne bi bilo brzo dostignuto**
  - Ukoliko su u korpusu **kraći dokumenti** (npr. članci)
    - Traženi **termini teže manje puta da se pojavljuju u dokumentu**, iako su **relevantni** u konktekstu upita
    - Zadaje se **manja vrednost parametra *k*,** kako **bi zasićenje bilo brzo dostignuto**

BM25 – podešavanje parametara *k* i *b*

- Kako odabrati **vrednost parametra *b*?** Pitanje je kakav je **domen dokumenata** – odnosno da li obimni dokumenti loše utiču na relevantnost termina
  - U slučaju **veoma specifičnih ili formalnih dokumentata** u kojima je svaka reč važna i doprinosi razumevanju dokumenta
    - Poput patenata, inženjerskih, medicinskih, pravnih ili naučnih dokumenata
    - Teži se **manjom vrednošću parametra *b***, jer **dužina dokumenta ne mora biti veoma relevantna**
  - U slučaju **manje specifičnih i formalnih dokumenata** koji pokrivaju **širi spektar tema**
    - Poput novina, recenzija proizvoda ili postova
    - Teži se **većom vrednošću parametra *b***, kako bi se **smanjio uticaj većeg broja pojavljivanja nerelevantnih termina ili kompletnih tema**

BM25 – ocena relevantnosti za upit

- Poput TF-IDF, **rezultat BM25** koristi se kao **ocena relevantnosti dokumenata** na osnovu **termina zadatih u upitu**
  - Koristi se za potrebe **rangiranja dokumenata** i vraćanja **najrelevantnijih rezultata**
  - Razlika je u **modifikaciji TF i IDF**, kao i upotrebi **parametara *k* i *b***

$$s(Q,d,k,D,b) = \sum_{i=1}^{n} BM25(q_i,d,k,D,b) = \sum_{i=1}^{n} TF_{BM25}(q_i,d,k,D,b) * IDF_{BM25}(q_i,D)$$

Q – termini postavljenog upita,
q – pojedinačni termin upita,
n – broj termina u upitu,
D – skup svih dokumenata u indeksu,
d – dokument za koji se računa ocena relevantnosti svih termina iz upita,
k – faktor nelinearnog zasićenja učestalosti termina,
b – faktor normalizacije učestalosti termina zasnovan na dužini dokumenta

TF-IDF i BM25 – faktor pojačavanja relevantnosti

- Ukoliko se rezultat upita doprema iz **različitih izvora** (npr. iz različitih indeksa ili različitih polja dokumenata), moguće je određenim **izvorima povećati relevantnost** (engl. *Boosting*) prilikom rangiranja dokumenata
  - Prilikom računanja relevantnosti dokumenata, **ocena relevantnosti množi se sa faktorom pojačavanja** (engl. *Boost Factor – BF*)
    - **Ocena relevantnosti dokumenata** računa se množenjem sledeće **tri komponente: TF, IDF i BF**
  - Čime se određenim **izvorima daje prioritet**
  - Npr. veću relevantnost moguće je dati poljima koja predstavljaju naslove dokumenata nego pasusima (ili primer naslova i opisa proizvoda), jer time je veća verovatnoća da se ceo dokument odnosi na postavljen upit
- Faktor pojačavanja **ne filtrira dokumente**, već samo **menja njihovu ocenu, odnosno rang**
- Osim konkretnih **numeričkih vrednosti** faktora pojačavanja, moguće je dodati i različite **funkcije** za računanje faktora pojačavanja
  - Npr. dokumenti novijeg datuma ili određene geolokacije imaju veću relevantnost

TF-IDF i BM25 – faktor pojačavanja relevantnosti

- **Faktor pojačavanja** moguće je **definisati na nivou:**
  - **Polja** – različita polja dokumenata imaju različitu relevantnost
    - Npr. naslov, opis, recenzije proizvoda
  - **Indeksa** – različiti indeksi (kolekcije) dokumenata imaju različitu relevantnost
    - Npr. indeks novih proizvoda, indeks popularnih proizvoda i indeks zastarelih proizvoda
  - **Upita** – različiti podupiti mogu imati različitu relevantost
    - Npr. upotreba istih ili različitih klauzula unutar upita, sa različitim faktorima pojačavanja
  - **Termina** – moguće je pojačati relevantnost konkretnih termina u upitu
    - Npr. ukoliko su određene reči važnije za pretragu
- **Faktor pojačavanja** moguće je **postaviti:**
  - **Trajno**, za bilo koji postavljen upit
    - Modifikovane ocene dokumenata skladište se u **invertovanom indeksu** – retko se koristi
  - **Privremeno**, za konkretan jedan upit

TF-IDF i BM25 – dodatne napomene

- **Ocene relevantnosti termina** moguće je izračunati i **skladištiti u invertovanom indeksu** prilikom **upisa dokumenata**
  - Dok se **ocena relevantnosti dokumenata na osnovu upita** i njihovo rangiranje sprovodi prilikom **izvršavanja upita**
  - Na taj način postiže se **kraće vreme izvršavanja upita**
- **Varijacija algoritma BM25** koju koristi *Elasticsearch*, odnosno biblioteka *Lucene*, **množi komponentu TF sa k+1**, kako bi približila vrednosti rezultata algoritma BM25 drugim algoritmima i potencijalno omogućila njihovo **jednostavnije poređenje**
  - Kako se koristi konstantna vrednost za sve dokumente, ona **nema uticaj na rangiranje dokumenata**

$$TF_{BM25}(t, d, k, D, b) = \frac{(k+1) * freq(t,d)}{freq(t,d) + k * \left(1 - b + b * \frac{count(d)}{avg(D)}\right)}$$

TF-IDF i BM25 – dodatne napomene

- Postoje **različiti algoritmi ocene relevantnosti dokumenata**, ali ne može se tvrditi da je **jedan algoritam uvek bolji od drugog**
- **Poređenje TF-IDF i BM25:**
  - **TF-IDF** jednostavniji, manje resursno zahtevan, dobar nad manjim skupom dokumenata, ali može da bude lošiji nad velikim skupom dokumenata
  - **BM25** kompleksniji, više resursno zahtevan, precizniji, dobar nad većim skupom dokumenata, nudi parametre za fina podešavanja

TF-IDF i BM25 – dodatne napomene

- **TF-IDF** nastao je kada je **uklanjanje zaustavnih (stop) reči bilo uobičajeno**
  - Stoga je bilo **prihvatljivo što komponenta TF ima linearan rast**, jer su učestale reči bile izbačene
  - Međutim, iako zaustavne reči imaju malu relevantnost i dalje **postoji određeni značaj tih reči**
    - Takođe, **zaustavne reči zavise i od jezika i od domena dokumenata** – za određene specifične dokumente može da važi drugačiji skup zaustavnih reči
  - Danas se neretko **zadržavaju zaustavne reči** prilikom procesiranja teksta, ali tada TF-IDF nije više pogodan
    - **BM25** uveden je kako bi se **zasićenjem termina** rešio problem prilikom zadržavanja **zaustavnih reči i ostalih učestalih reči**

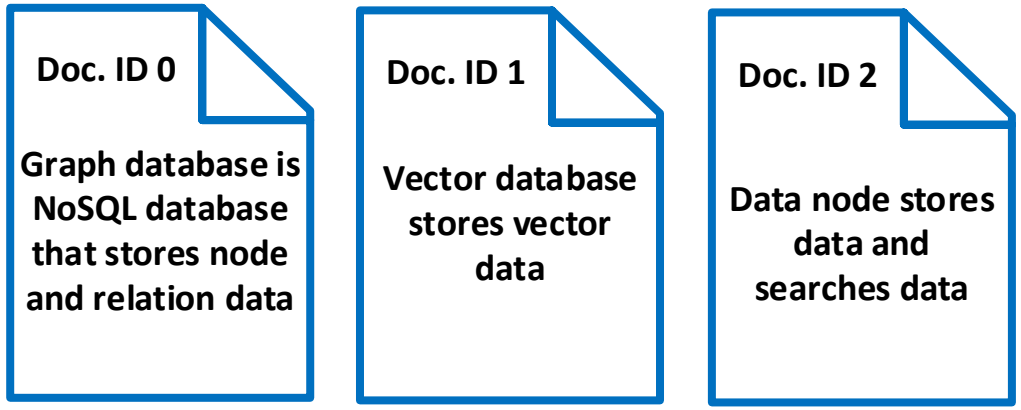
TF-IDF i BM25 – poboljšanja rezultata pretrage

- Moguće opcije za **poboljšanja kvaliteta dobijenih rezultata pretrage:**
  - Upotreba različitih komponenti za **procesiranje teksta:**
    - **Definisanje i upotreba sinonima i akronima** prilikom indeksiranja dokumenata i postavljanja upita (engl. *Synonyms*)
    - Svođenje na **koren reči ili lemu** (engl. *Stemming / Lemmatization*)
    - Upotreba **tolerancije na slovne greške** prilikom postavljanja upita (engl. *Fuzziness*)
    - Omogućavanje **predloga reči prilikom postavljanja upita** (engl. *Typeahead/Autocomplete*)
    - Podrška **različitim izgovorima** (engl. *Phonetic Matching*)
  - Dodavanje **faktora pojačavanja** u slučaju korišćenja različitih izvora
  - **Modifikacija ocene relevantnosti** rezultata upita od strane **kreiranih funkcija na osnovu dodatnih faktora**
    - Npr. smanjiti ocenu relevantnosti starijih dokumenata – novi dokumenti imaju veću ocenu, dok stari dokumenti imaju manju ocenu
    - Npr. smanjiti ocenu relevantnosti dokumenata koji su geografski udaljeniji od korisnika – tražene prodavnice u gradu koje su blizu korisnika imaju veću ocenu, dok udaljene prodavnice imaju manju ocenu
  - Kako podrazumevane **vrednosti parametara *k* i *b* algoritma BM25** uglavnom daju dobre rezultate, najpre se testiraju ostale opcije pre podešavanja navedenih parametara

TF-IDF i BM25 – pretraga sličnosti

- Moguće je pojedinačne **dokumente posmatrati kao vektore**
  - **Dimenzije** vektorskog prostora reprezentuju **termine**
    - Vrednosti dokumenata za svaku **dimenziju** predstavljaju **ocene relevnatnosti termina** dobijenih od algoritma poput TF-IDF ili BM25
  - Moguće je iskoristiti **metriku rastojanja dva vektora** kako bi bili pronađeni slični dokumenti
    - Npr. u sistemima preporuka ili za klasifikaciju dokumenata
    - U tom slučaju, **pozicija termina u tekstu nema uticaj** na rezultat pretrage
- Međutim, i u vektorskim bazama podataka, prilikom hibridne pretrage, **najčešće se računa ocena relevantnosti dokumenata** prilikom postavljanja upita, umesto upotrebe metrike sličnosti za retke vektore

TF-IDF i BM25 – pretraga sličnosti



Napomena: vrednosti su navedene kao primer, ne predstavljaju realne vrednosti TF-IDF ili BM25

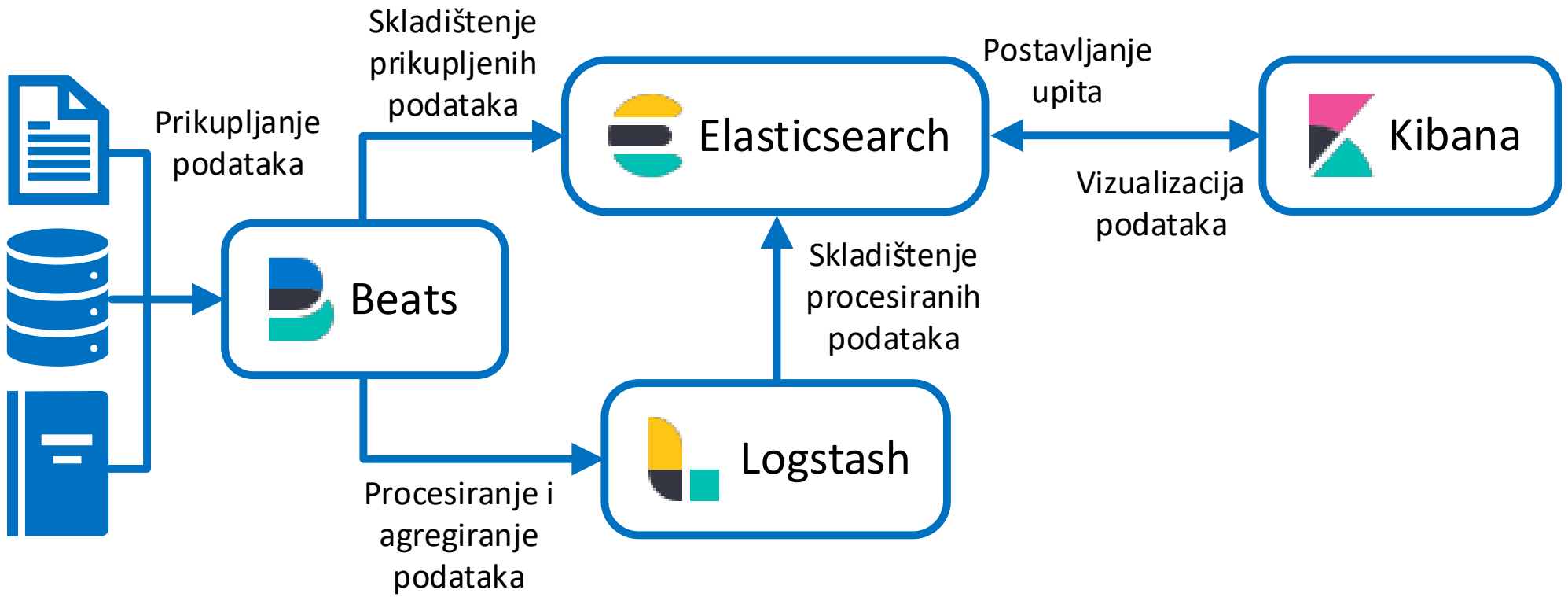
	and	data	database	graph	is	node	nosql	relation	searches	stores	that	vector
D0	0,251	0,195	0,502	0,330	0,330	0,251	0,330	0,330	0	0,195	0,330	0
D1	0	0,257	0,331	0	0	0	0	0	0	0,257	0	0,871
D2	0,320	0,746	0	0	0	0,320	0	0	0,421	0,249	0	0

Sadržaj

Elastic Stack

- ***Elastic Stack*** predstavlja **platformu za pretraživanje** koju čini **skup alata** za potrebe **prikupljanja, obrade, skladištenja, pretrage, analize i vizualizacije** velike količine podataka
  - Nastao je inicijalno kao potreba za **jednostavnim učitavanjem i vizualizacijom logova**
  - ***Elasticsearch*** deo je platforme ***Elastic Stack*** i predstavlja njenu **glavnu komponentu**
- *Elastic Stack* obuhvata sledeće **alate**:
  - ***Beats*** – prikupljanje podataka
  - ***Logstash*** – procesiranje i agregiranje prikupljenih podataka
  - ***Elasticsearch*** – indeksiranje, skladištenje i pretraga podataka
  - ***Kibana*** – vizualizacija podataka

Elastic Stack



Izvor ikonica alata sistema Elastic Stack: https://www.elastic.co/

Beats

- ***Beats*** predstavlja **kolekciju agenata** koji **prikupljaju podatke** sa **različitih izvora**, poput sistema, uređaja, mašina
  - Šalju prikupljene podatke u ***Logstash* na procesiranje ili direktno u *Elasticsearch* na skladištenje**
  - Implementirani su da zauzimaju **malo računarskih resursa** i da budu **jednostavni za instalaciju i konfigurisanje**
  - **Primeri *Beats* agenata:**
    - *Filebeat* – prikuplja log datoteke
    - *Metricbeat* – prikuplja metrike sistema
    - *Packetbeat* – prikuplja podatke mrežnog saobraćaja
    - *Winlogbeat* – prikuplja logove događaja sa operativnog sistema *Windows*
    - *Auditbeat* – prikuplja logove korisničkih aktivnosti i procesa
    - *Heartbeat* – prikuplja podatke dostupnosti servisa
  - Npr. mogu da budu pokrenuti na serveru kako bi pratili log datoteke, parsirali ih i slali u *Logstash* ili *Elasticsearch*

Logstash

- ***Logstash*** predstavlja **alat za agregiranje i procesiranje podataka**
  - Vrši procesiranje prikupljenih, **neprocesiranih podataka** (engl. *Raw Data*)
  - **Transformiše podatke** različitih formata iz različitih izvora u **odgovarajući format**
  - Vrši **prikupljanje, parsiranje, transformisanje i agregiranje** podataka
    - Predstavlja **pogon ETL** (engl. *Extract, Transform, Load* (*ETL*) *Engine*)
  - **Pripremljene podatke** prosleđuje u *Elasticsearch*

Elasticsearch

- ***Elasticsearch*** predstavlja **bazu podataka i pogon za pretraživanje** koji služi prvenstveno za **punu pretragu teksta**
  - **Indeksira, skladišti i pretražuje podatke** dobijene iz *Beats* ili *Logstash*
  - Obezbeđuje jednostavne **REST pristupne tačke** za pristup bazi podataka
  - Koristi se često i termin **pogon za pretraživanje i analizu** (engl. *Search and Analytics Engine*)
- Razvijen je u programskom jeziku **Java** i zasnovan na biblioteci ***Apache Lucene***
  - ***Lucene Core*** predstavlja Java biblioteku koja omogućava različite vrste **pretraga i analize teksta, indeksiranje, provere pravopisa i brzu identifikaciju važnih reči ili fraza**
  - Omogućava razvoj efikasnih i preciznih **pretraga tekstualnih datoteka**
- *Elasticsearch* čuva podatke u formi **dokumenata JSON**
  - Time se može predstaviti kao **baza podataka orijentisana ka dokumentima**
  - Može da radi sa strukturiranim, polustrukturiranim i nestrukturiranim podacima

JSON

- **JSON** (engl. *JavaScript Object Notation*) predstavlja **format za razmenu i deljenje podataka** između različitih sistema ili aplikacija, najčešće **klijenta i servera**
  - Može da služi i kao **format skladištenja podataka** u bazi podataka
  - Jednostavan je za **čitanje i razumevanje** od strane čoveka
- JSON predstavlja **strukturiran format podataka** zasnovan na **sintaksi jezika *JavaScript***, što ga čini jednostavnim za konvertovanje u *JavaScript* objekat
- Kreiran je da bude korišćen **nezavisno od odabira programskog jezika**
  - Moguće je koristiti format JSON u **raznim programskim jezicima i konvertovati ga u objekte** odabranog jezika
  - Što ga čini pogodnim za **razmenu podataka između sistema napisanih u različitim jezicima** (npr. *C#*, *Java*, *Python*)

Sintaksa JSON

- **Objekat JSON** sastavljen je od sledećih osnovnih elemenata:
  - Okružen je **vitičastim zagradama**
  - Čini ga **skup parova ključ-vrednost**, u kojoj je vrednost razdvojena od ključa dvotačkom, dok je svaki par razdvojen zarezom
  - **Ključ** mora da bude tipa *String*, dok **vrednost** može da bude jedan od različitih tipova kao što su *String*, broj, objekat i *Boolean*
- Moguće je **transformisati** objekat JSON u objekat **drugih formata** poput XML (engl. *Extensible Markup Language*) i CSV (engl. *Comma-Separated Values*)

Sintaksa JSON

- **Primer** objekta student u formatu JSON:

```
{
    "ImePrezime": "Jovana Jovanović",
    "BrojIndeksa": {
        "StudijskiProgram": "RN",
        "Broj": 150,
        "GodinaUpisa": 2020
    },
    "PoloženiIspiti": ["Baze podataka 1", "Baze podataka 2", "Sistemi baza podataka"],
    "GodinaStudija": 4,
    "PoloženaStručnaPraksa": false,
    "Stranica": null
}
```

Kibana

- ***Kibana*** predstavlja korisnički interfejs za **vizualizaciju podataka i upravljanje sistemom *Elasticsearch***
  - Omogućava **uvid, pretragu, analizu i vizualizaciju podataka** skladištenih u bazi podataka *Elasticsearch* pomoću različitih grafikona
  - Omogućava kreiranje **interaktivnih kontrolnih tabli** (engl. *Dashboards*), kao i **praćenje i upravljanje sistemom *Elastic Stack***, odnosno različitih **metrika i trendova**
  - Moguće je umesto alata *Kibana* implementirati **sopstveni alat** za vizualizaciju podataka ili koristiti **druge postojeće alate**

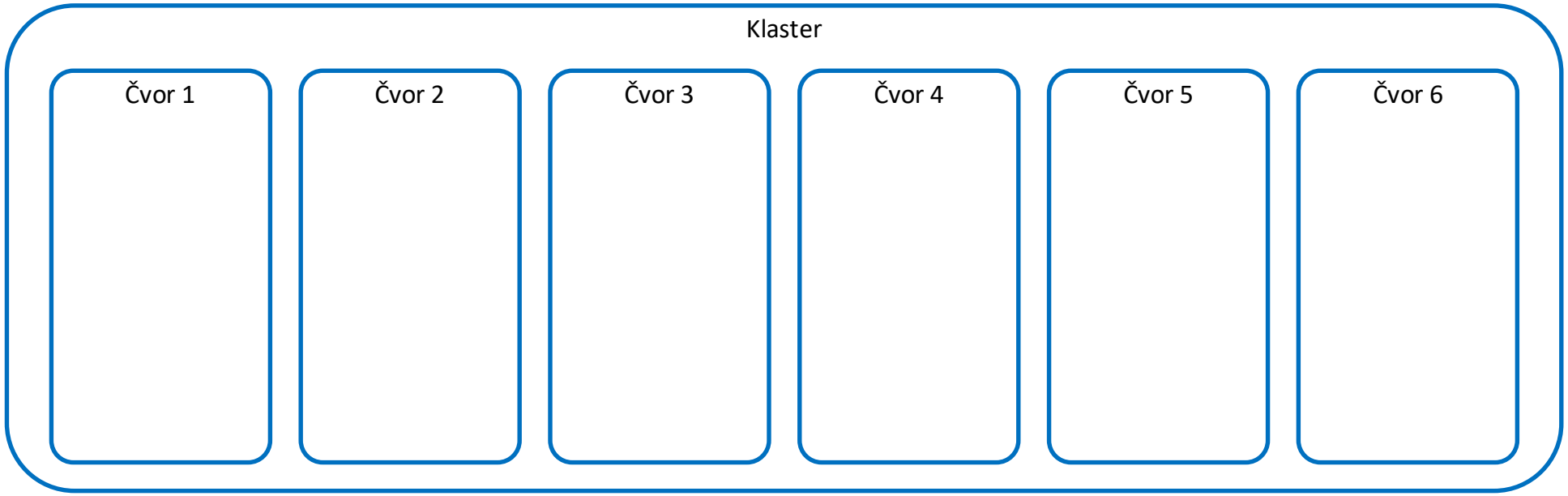
Sadržaj

Arhitektura Elasticsearch

- **Osnovni koncepti arhitekture** baze podataka i pogona za pretraživanje ***Elasticsearch*:**
  - **(A) Klaster** (engl. *Cluster*)
  - **(B) Čvor** (engl. *Node*)
  - **(C) Indeks** (engl. *Index*)
  - **(D) Dokument** (engl. *Document*)
  - **(E) Fragment** (engl. *Shard*)
  - **(F) Replika** (engl. *Replica*)

Arhitektura Elasticsearch – klaster

- **(A) Klaster** predstavlja **kolekciju čvorova**, odnosno servera
  - Može da sadrži **jedan ili više čvorova**
  - Čvorovi u okviru klastera se koriste kako bi bila izvršena **distribucija različitih zadataka, pretraga i indeksiranja**, odnosno **raspodela opterećenja**
  - Svaki čvor unutar klastera može **pristupiti ostalim čvorovima** istog klastera



Arhitektura Elasticsearch – čvor

- **(B) Čvor** predstavlja pojedinačnu **instancu servera**
  - Predstavljaju pripadnike klastera koji **dele isti cilj**
  - Učestvuju u **indeksiranju** unutar klastera, kao i za potrebe **pretraga** unutar klastera
  - Na osnovu dodeljenih zaduženja, **čvorove je moguće podeliti na sledeće osnovne uloge:**
    - **Čvor prikupljanja podataka** (engl. *Ingest Node*) – prikuplja i preprocesira podatke
    - **Čvor podataka** (engl. *Data Node*) – indeksira, skladišti, pretražuje i agregira podatke
    - **Čvor koordinacije** (engl. *Coordinating-Only Node*) – uspostavlja komunikaciju sa krajnjim korisnikom
    - **Čvor upravljanja** (engl. *Master Node*) – upravlja i održava klaster
  - Pored osnovnih, postoje i sledeće uloge čvorova:
    - **Čvor mašinskog učenja** (engl. *Machine Learning Node*) – izvršava algoritme mašinskog učenja
      - Npr. za potrebe detekcije anomalija ili predikcija
    - **Čvor transformisanja** (engl. *Transform Node*) – agregira i transformiše podatke, odnosno transformiše postojeće indekse u nove indekse sa agregiranim podacima za potrebe analize podataka
  - Moguće je da čvor **deli više uloga**

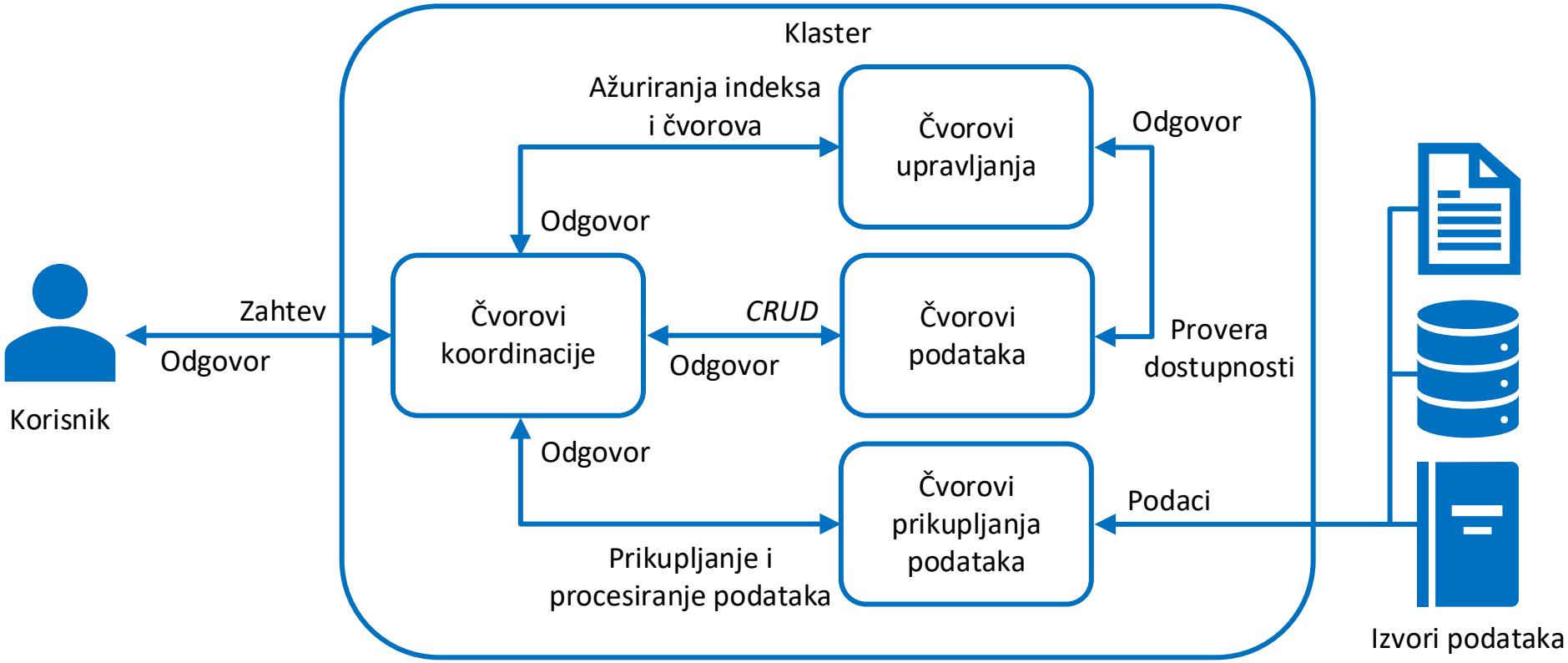
Arhitektura Elasticsearch – osnovni tipovi čvorova

- **Čvor prikupljanja podataka** zadužen je za **prikupljanje i preprocesiranje podataka**
  - Vrši ekstrakciju podataka iz **različitih izvora** i transformiše podatke u **odgovarajući format**
    - Npr. parsiranje log datoteka; ekstrakcija podataka iz polja; parsiranje dokumenata
  - **Smanjuje opterećenje** čvorova podataka i koordinacije time što preuzima zaduženja prikupljanje podataka
- **Čvor podataka** zadužen je za **indeksiranje i skladištenje podatke, osnovne *CRUD* operacije, i pretraživanje i agregiranje podataka**
  - Navedene operacije mogu zahtevati **znatne računarske resurse**
    - Poželjno je čvor podataka **odvojiti od ostalih čvorova u klasteru**
  - Koristi se za **ažuriranje podataka i postavljanje upita**
  - Podaci su skladišteni u **fragmentima**

Arhitektura Elasticsearch – osnovni tipovi čvorova

- **Čvor koordinacije** zadužen je za **komunikaciju sa krajnjim korisnikom**
  - **Prikuplja zahteve** od strane korisnika i prosleđuje ih ostalim čvorovima
  - U slučaju da se relevantni dokumenti nalaze u **više različitih čvorova podataka**, čvor koordinacije:
    - **Prima upit** od strane korisnika
    - **Distribuira upit** različitim čvorovima podataka koji obuhvataju fragmente sa relevantnim dokumentima
    - **Prikuplja rezultate** upita iz različitih čvorova podataka i agregira ih u jedan rezultat
    - **Vraća formirani rezultat** korisniku
  - Omogućava **bolje performanse i skalabilnost** celokupnog sistema
- **Čvor upravljanja** zadužen je za **administrativne zadatke unutar klastera**
  - Vrši **proveru dostupnosti** čvorova podataka
  - Koristi se za **kreiranje i uklanjanje indeksa i čvorova**
  - Prati **stanje** i konfiguriše **podešavanja klastera**, poput preslikavanja indeksa i alokacija fragmenata
  - Kako administrativni zadaci nisu zahtevni, dovoljno je postojanje **jednog čvora upravljanja**
    - U slučaju njegovog **otkaza**, privremeno se uloga upravljanja dodeljuje nekom od čvorova u klasteru

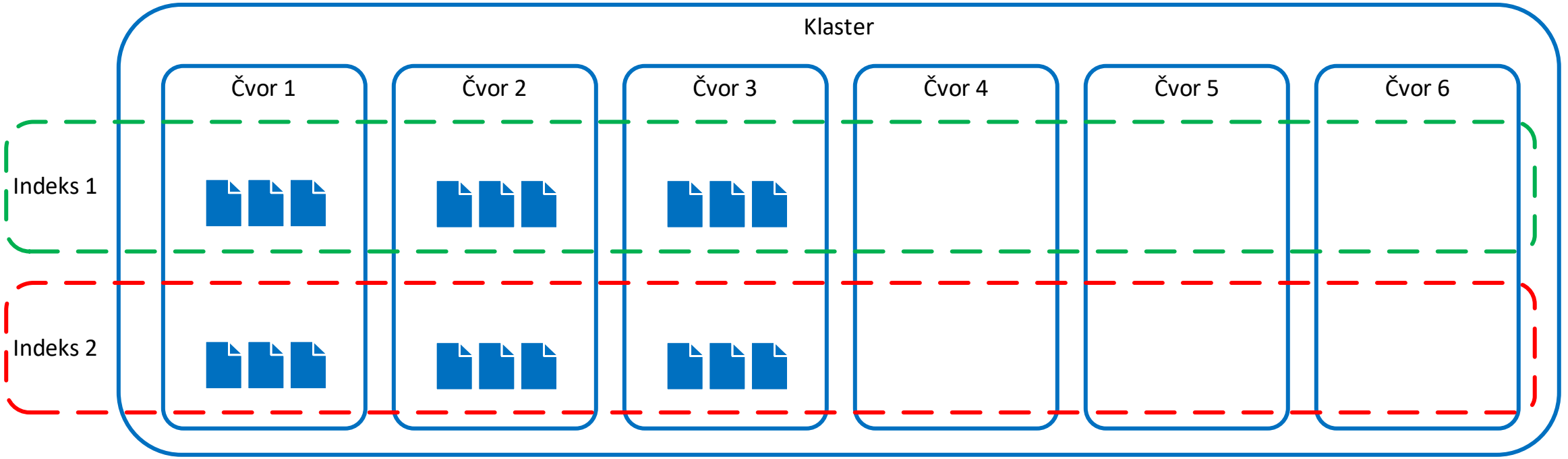
Arhitektura Elasticsearch – osnovni tipovi čvorova



Arhitektura Elasticsearch – indeks

- **(C) Indeks** predstavlja **kolekciju srodnih dokumenata** koji imaju sličnu strukturu
  - **Logički grupiše dokumente** koji dele sličnu temu ili su slične semantike
    - **Ne organizuje** dokumente **fizički** u masovnoj memoriji
  - Moguće je kreirati različite indekse za **različite tipove entiteta**
    - Čime se **sužava prostor pretrage** relevantnih dokumenata
    - Npr. internet prodavnica poseduje indeks *Proizvod* i indeks *Korisnik*

Arhitektura Elasticsearch – indeks



Arhitektura Elasticsearch – tip preslikavanja

- **Tip preslikavanja** (engl. *Mapping Type*) predstavljao je **tip dokumenata ili entiteta u okviru indeksa**
  - Npr. unutar indeksa *Proizvod* postoji tip *Računar* i tip *Bela tehnika*
  - Korišćen u *Elasticsearch* **do verzije 8**, nakon čega je **potpuno uklonjen**
    - Razlog – skladištenje različitih tipova entiteta koji imaju **mali broj zajedničkih polja** unutar jednog indeksa dovodi do **retkih podataka i onemogućava dobru kompresiju podataka**
    - Takođe, **polja sa istim nazivom** u različitim tipovima preslikavanja skladištena su kao **jedno polje** – nije bilo moguće imati **različite tipove podataka za isti naziv polja** u različitim tipovima preslikavanja
  - **Umesto upotrebe tipova**, moguće je:
    - **Kreirati različite indekse** za kolekcije dokumenata koje reprezentuju različite tipove entiteta
      - Bolje **indeksiranje teksta** (TF), kao i **kompresija podataka** jer su srodni entiteti obuhvaćeni indeksom
      - Npr. kreirati indeks *Računar* i indeks *Bela tehnika*
    - **Kreirati polje** po kojem će se dokumenti razlikovati
      - Pogodno za **relativno mali broj dokumenata** – da se ne pravi poseban indeks
      - Npr. indeks *Proizvod* ima polje *Kategorija*

Arhitektura Elasticsearch – dokument

- **(D) Dokument** predstavlja **osnovnu jedinicu skladištenja**, odnosno jedan slog
  - Skladišti se u **formatu JSON i sadrži kolekciju polja** tipa ključ-vrednost
  - **Ne postoji jasno definisana šema dokumenata** u okviru baze podataka *Elasticsearch*
  - Međutim, moguće je kreirati **preslikavanja** koja predstavljaju **kolekciju naredbi** datih *Elasticsearch* u vezi **načina čuvanja i prikupljanja različitih podataka**
    - Moguće je **definisati polja** u indeksu, odrediti njihove **tipove podataka**, kao i **način korišćenja** polja od strane *Elasticsearch*
    - Npr. moguće je označiti polja koja će se koristiti za punu pretragu teksta (engl. *Full-Text Search*)
    - Ukoliko **preslikavanja nisu definisana**, *Elasticsearch* će pokušati da identifikuje strukturu polja koristeći **dinamičko preslikavanje**
  - **Meta-polja** (engl. *Meta-Fields*) skladište **dodatne informacije o dokumentima** koje *Elasticsearch* interno koristi prilikom pristupanja dokumentima
    - Npr. naziv indeksa u kojem je dokument skladišten; jedinstveni identifikator dokumenta

Arhitektura Elasticsearch – tipovi podataka

- Postoje različiti **tipovi podataka** koji mogu biti dodeljeni poljima dokumenata
  - **Tekst** – koristi se za **punu pretragu teksta** (npr. opis proizvoda)
    - Vrši se analiza, tokenizacija i indeksiranje nestrukturiranog tekstualnog sadržaja
    - Nije moguće vršiti uređivanje i agregiranje tekstualnih podataka
  - **Ključne reči** – koristi se za pretragu sa **tačnim poklapanjem teksta** (npr. e-adresa korisnika)
    - Može se koristiti za filtriranje, agregiranje i uređivanje rezultata pretrage
    - Ne vrši se analiza i tokenizacija tekstualnog sadržaja – ne menja se sadržaj
    - Ne koriste se za punu pretragu teksta
  - **Numerički** – tipovi podataka poput *integer*, *long*, *short*, *float* i *double*
    - Indeksiraju se na način da omoguće efikasne **pretrage opsega i agregacije**
  - **Datumski** – tip podataka za datum i vreme
    - Omogućena su različita **formatiranja datuma i operacije** nad njima
  - ...

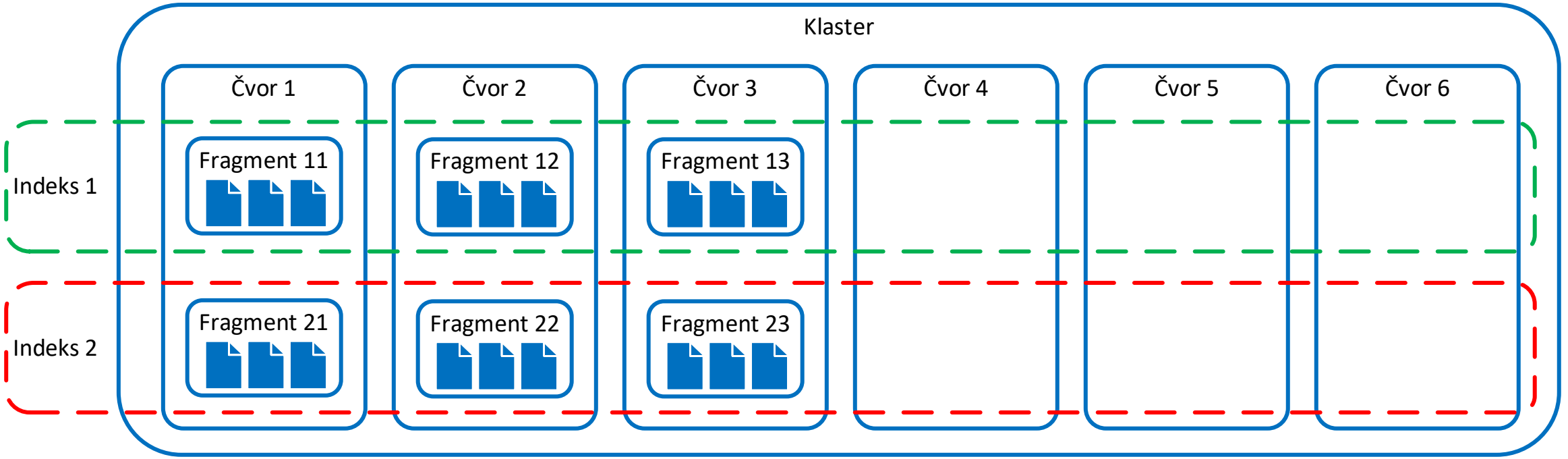
Arhitektura Elasticsearch – tipovi podataka

- Postoje različiti **tipovi podataka** koji mogu biti dodeljeni poljima dokumenata
  - …
  - **Tačka u prostoru** – geoprostorni podaci
    - Koriste se za **skladištenje i pretragu lokacija** na osnovu geografske širine i dužine (npr. urediti prodavnice na osnovu blizine lokacije)
  - **Objekat** – objekti JSON
    - Koristi se za skladištenje **kolekcije polja i vrednosti**
  - **Ugnježđeni objekat** – biblioteka *Lucene* ne podržava korišćenje niza objekata
    - Postoji poseban tip podatka *Nested* – **čuva niz objekata kao posebne dokumente**
  - **Ostali tipovi podataka** – postoje razni drugi tipovi podataka poput *boolean* i *IP address*

Arhitektura Elasticsearch – fragment

- **(E) Fragment** predstavlja **podskup dokumenata određenog indeksa**
  - Služi za **fizičku organizaciju** dokumenata po čvorovima
  - Omogućava **horizontalno skaliranje**
    - Indeks može da skladišti **velik broj dokumenata**, što utiče na performanse i održavanje
    - Moguće je **podeliti dokumente indeksa** na više delova, distribuirati delove u **različite čvorove** i omogućiti **smanjenje vremena izvršavanja upita** primenom **paralelnog** izvršavanja
    - Moguće **je jednostavno dodati nove čvorove i fragmente** kako broj dokumenata raste
  - Fragmenti mogu biti **skladišteni u istom ili različitim čvorovima** klastera
  - Prilikom kreiranja indeksa potrebno je **definisati broj fragmenata**
    - Mora postojati **barem jedan** fragment za svaki indeks
    - Ukoliko je potrebno **promeniti broj fragmenata**, neophodno je **ponovo kreirati indeks**

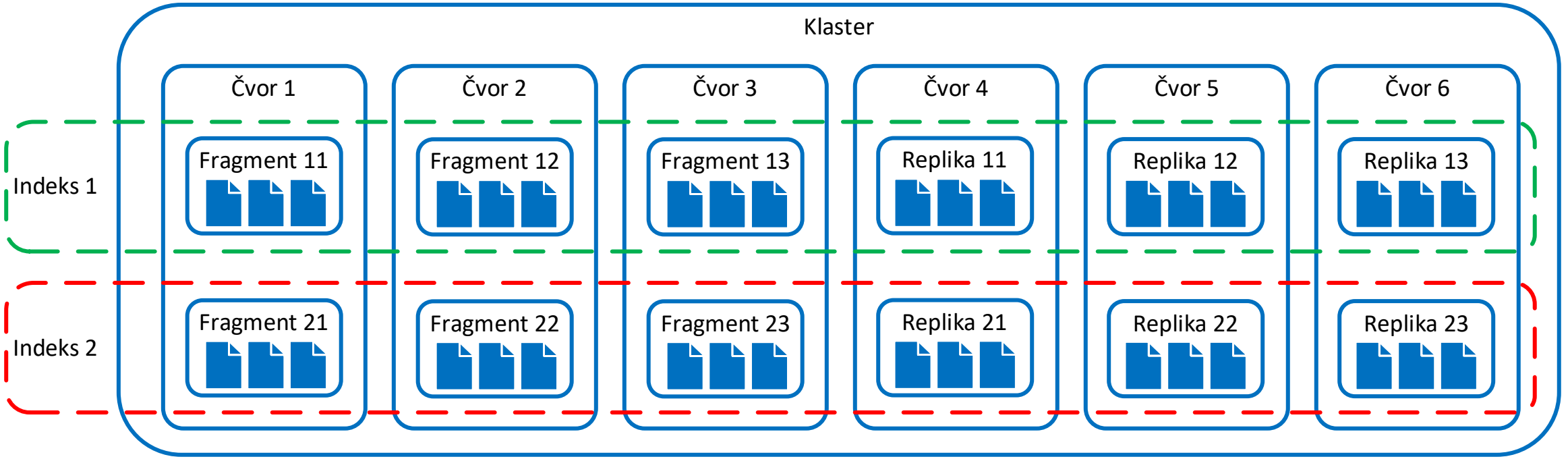
Arhitektura Elasticsearch – fragment



Arhitektura Elasticsearch – replika

- **(F) Replika** predstavlja **kopiju fragmenta**
  - Često se originalni fragment naziva **primarni fragment** (engl. *Primary Shard*), a njena kopija se naziva **replika fragmenta** (engl. *Replica Shard*) ili samo **replika**
  - Skladišti se u čvorovima u kojima se **ne nalazi odgovarajući primarni fragment**
  - Obezbeđuje **otpornost na otkaze i visoku dostupnost sistema**
    - Omogućuje nastavak rada sistema u slučaju **otkaza jednog ili više čvorova**
  - Moguće da **poboljša performanse izvršavanja upita** u slučaju velikog broja korisnika
    - **Raspodela upita na različite čvorove** koji obuhvataju replike primarnih fragmenata

Arhitektura Elasticsearch – replika



Jezici za rad sa bazom podataka Elasticsearch

- Moguće je koristiti **postojeće programske jezike** za pristup bazi podataka *Elasticsearch* (npr. *Java*, *JavaScript*, *C#*, *Python*, *Go*)
- *Elasticsearch* koristi jezik ***Query DSL***
  - **Namenski jezik** (engl. *Domain-Specific Langauge*) za pristup bazi podataka *Elasticsearch*
  - Kreira upite **u formi API i JSON**
- Primer dodavanja proizvoda:

```
PUT proizvod/_doc/1
{
  “naziv”: “Uređaj 1”,
  “opis”: “Opis uređaja 1”
}
```

- Primer čitanja proizvoda:

```
GET proizvod/_doc/1
```

Tipovi upita

- *Elasticsearch* podržava različite **tipove upita**, poput:
  - ***Match Queries*** – Pretraga dokumenata na osnovu termina ili fraza u tekstu koja zahteva **analizu teksta i rangiranje rezultata pretrage**
    - Koristi **ocenu relevantnosti dokumenata** za rangiranje rezultata pretrage
  - ***Term Queries*** – Pretraga dokumenata na osnovu **zadatih termina ili fraza** u određenom polju
    - Ne koristi **analizu teksta i ocenu relevantnosti dokumenata**
  - ***Range Queries*** – Pretraga dokumenata čije vrednosti polja ulaze u zadati **opseg vrednosti**
    - Koriste se za **numeričke i datumske** tipove podataka
  - ***Prefix Queries*** – Pretraga dokumenata čije vrednosti polja **počinju zadatim tekstom**
    - Mogu biti korisni za **predlaganje reči** (engl. *Autocomplete*) ili za pretragu termina koji **dele isti prefiks**
  - ***Wildcard Queries*** – Pretraga dokumenata na osnovu zadatog **šablona**
    - * zamena za 0 ili više karaktera; ? zamena za tačno jedan karakter
  - ***Fuzzy Queries*** – Pretraga dokumenata na osnovu termina koji su **slični zadatom terminu**
    - Može da uzme u obzir manje **slovne greške prilikom kucanja teksta ili varijacije teksta**

Postupak izvršavanja upita u Elasticsearch

- **Upit za punu pretragu teksta u *Elasticsearch*** izvršava se pomoću sledećih koraka:
  - **Korak 1. Postavljanje upita** od strane korisnika
  - **Korak 2. Prihvatanje postavljenog upita** od strane čvora koordinacije
  - **Korak 3. Parsiranje upita** od strane čvora koordinacije, radi identifikacije tipa upita i parametara upita
  - **Korak 4. Identifikovanje fragmenata** koji sadrže relevantne dokumente i **distribuiranje upita** čvorovima podataka od strane čvora koordinacije
  - **Korak 5. Pretraga dokumenata i lokalno rangiranje** pronađenih dokumenata u svakom čvoru podataka posebno
  - **Korak 6. Spajanje rezultata pretraga** svakog čvora podataka od strane čvora koordinacije
  - **Korak 7. Globalno rangiranje svih pronađenih dokumenata** od strane čvora koordinacije
  - **Korak 8. Formatiranje i slanje rezultata pretrage** korisniku od strane čvora koordinacije
  - **Korak 9. Prezentovanje rezultata pretrage** u odgovarajućem formatu (npr. *Kibana*)

Karakteristike i funkcionalnosti Elasticsearch

- **Glavne karakteristike i funkcionalnosti** *Elasticsearch*:
  - Baza podataka **orijentisana ka dokumentima**
    - Skladišti dokumente u **formatu JSON**
  - **Brza i precizna pretraga** podataka u približno realnom vremenu
    - Obezbeđuje **distribuiranu pretragu**
    - Efikasno može da obradi **veliku količinu podataka**
    - Može da otkrije **trendove i šablone**
  - Efikasna **puna pretraga teksta**
    - Vrši pretragu **nestrukturiranog tekstualnog sadržaja**
    - Može da **rangira** pronađene dokumente po **relevantnosti**
  - **Jednostavno skaliranje** podataka kroz distribuiranu arhitekturu *Elasticsearch*
    - Moguće **horizontalno i vertikalno** skaliranje podataka
  - **Analiza i vizualizacija** podataka
    - Vizualizacija podataka u formi grafikona moguća je uz pomoć alata ***Kibana***

Prednosti i nedostaci Elasticsearch

- **Prednosti**:
  - Mogućnost **pune pretrage teksta**
  - **Jednostavan pristup** usled upotrebe REST pristupnih tačaka
  - **Fleksibilnost upisa podataka** usled nepostojanja šeme baze podataka
  - **Visoke performanse izvršavanja upita**
    - Posledica primene **invertovanih indeksa**, **distribucije čvorova** i izvršavanja upita u **paraleli**
  - Primena različitih alata za podršku **prikupljanju, skladištenju, analizi i vizualizaciji podataka**
- **Nedostaci**:
  - **Nepostojanje šeme baze podataka** može dovesti do nekonzistentnih podataka
  - Ne podržava **transakcionu obradu** podataka
  - Neophodna **instalacija Java virtuelne mašine** (engl. *Java Virtual Machine* (*JVM*))
  - Neophodno **rezervisati dovoljnu količinu radne memorije** za *Elasticsearch*

Primena Elasticsearch

- **Primeri primene:**
  - **Pretraga i analiza podataka** informacionog sistema
    - Npr. pretraga i analiza proizvoda ili poslovanja
    - Npr. pretraga dokumenata (nestrukturiranog teksta) u kompanijama
    - Moguće je **primeniti različite algoritme mašinskog učenja** za potrebe analize podataka
  - Obezbeđivanje funkcionalnosti **predlaganja reči** (engl. *Autocomplete*) korisniku prilikom navođenja reči pretrage u informacionom sistemu
  - **Skladištenje i analiza logova** (npr. praćenje rada i performansi sistema), **merenja** (npr. podaci sa senzora) ili **bezbednosnih događaja** (npr. detekcija anomalija)
  - **Vizualizacija i prikaz** rezultata ili trendova (engl. *Dashboard*)
  - Eksterna memorija u **arhitekturi RAG**
  - Pretraga **geoprostornih** podataka
    - Npr. pronalazak najbližih prodavnica
  - **Prikupljanje** (engl. *Scraping*) **i pretraga** podataka sa različitih **internet stranica**
    - Npr. objava ili blogova

Sadržaj

Poređenje relacione i baze podataka Elasticsearch

- *Elasticsearch* najčešće se koristi kao **sekundarna, pomoćna baza podataka**
  - **Iz primarne baze podataka** mogu se preneti **samo neophodni podaci** u bazu podataka *Elasticsearch*
    - Npr. identifikator i opis proizvoda, gde bi se tekst opisa proizvoda koristio za pretragu i analizu
  - Kao **primarna baza podataka** može se koristiti relaciona baza podataka
- Za razliku od relacione baze podatka, *Elasticsearch* čuva podatke u formi **dokumenata tipa JSON**
  - Predstavlja **denormalizaciju** podataka
  - **Nema šemu baze podataka**
  - **Ne podržava transakcije, ograničenja referencijalnih integriteta i podupite**

Poređenje relacione i baze podataka Elasticsearch

- **Koncepti** baze podataka *Elasticsearch* koji **mogu da se uporede** sa konceptima relacione baze podataka:

<table>
<tr><th>Baza podataka Elasticsearch</th><th>Relaciona baza podataka</th></tr>
<tr><td>Indeks
{
“id”: “101”,
“naziv”: “Uređaj 1”
}
{
“id”: “102”,
“naziv”: “Uređaj 2”
}</td><td>Relacija (tabela)
<table><tr><th>Id</th><th>Naziv</th></tr><tr><td>101</td><td>Uređaj 1</td></tr><tr><td>102</td><td>Uređaj 2</td></tr></table></td></tr>
<tr><td>Dokument
{
“id”: “101”,
“naziv”: “Uređaj 1”
}</td><td>N-torka (slog, red)
<table><tr><th>Id</th><th>Naziv</th></tr><tr><td>101</td><td>Uređaj 1</td></tr><tr><td>102</td><td>Uređaj 2</td></tr></table></td></tr>
<tr><td>Polje u dokumentu
{
“id”: “101”,
“naziv”: “Uređaj 1”
}</td><td>Obeležje (kolona)
<table><tr><th>Id</th><th>Naziv</th></tr><tr><td>101</td><td>Uređaj 1</td></tr><tr><td>102</td><td>Uređaj 2</td></tr></table></td></tr>
</table>

Poređenje relacione i baze podataka Elasticsearch

Baza podataka *Elasticsearch*	Relaciona baza podatka
Ne podržava šemu baze podataka	**Podržava šemu** baze podataka
Pogodna za **nestrukturirane, tekstualne podatke**	Pogodna za **strukturirane podatke**
Ne podržava transakcionu obradu podataka	**Podržava transakcionu obradu podataka**
Pogodna za **upite koji podrazumevaju punu pretragu i analizu teksta**	Pogodna za **upite sa kompleksnim uslovima pretrage**
Korisna za **skladištenje, pretragu i analizu velike količine podataka**	Korisna prilikom potrebe **očuvanja integriteta i konzistentnosti podataka**
Jednostavnije horizontalno skaliranje	**Izazovnije horizontalno skaliranje**

Sadržaj

Literatura

- Madhusudhan Konda, Elasticsearch in Action, 2nd Edition, Manning Publications Co., 2023.
- Clinton Gormley, Zachary Tong, Elasticsearch: The Definitive Guide: A Distributed Real-Time Search and Analytics Engine, 1st Edition, O'Reilly Media, Inc., 2015.

Napredne arhitekture informacionih sistema

Pogoni za pretraživanje

Pitanja?

Predmetni nastavnik:
dr Marko Vještica

DAIS
Data Science and Information Systems Group