

## Manje poznati izrazi

---

**Object-relational impedance mismatch** – greske koje nastaju kada se pokušava preneti objektno orijentisan model na relacioni model.

**CRUD** (**crud operations**) – obuhvataju operacije **C**reate, **R**ead, **U**ppdate, **D**eleate

**RESTFul**

**Binary safe** – niz bitova se posmatra samo kao niz bitova bez ikakvog značenja kao tekst.

**failover** – ukoliko dođe do greske u sistemu koja je fatalna postoji protokol koji vrši transfer kontrole i sistem nastavlja sa radom.

**in-memory** – obično se odnosi na radnu memoriju (ram). Nesto se nalazi u ram-u.

## 1. Karakteristike podataka na Web-u. Primeri struktuiranih i polustruktuiranih podataka.

- Velika količina podataka
- Povezanost podataka (relacije)
- Polustruktuiranost podataka
- Arhitektura aplikacija koje koriste podatke

### ▣ Struktuirani podaci

- Tradicionalni podaci u relacionim bazama podataka

Structured data first depends on creating a data model – a model of the types of business data that will be recorded and how they will be stored, processed and accessed. This includes defining what fields of data will be stored and how that data will be stored: data type (numeric, currency, alphabetic, name, date, address) and any restrictions on the data input (number of characters; restricted to certain terms such as Mr., Ms. or Dr.; M or F).

### ▣ Polustruktuirani i nestruktuirani podaci

- Format podataka nije konzistentan
- Veoma je teško definisati record/slog u tradicionalnom smislu
- Dominantan tip podataka na Web-u
- Zahtevaju novi pristup i nove tehnike za obradu

Semi-structured data is information that doesn't reside in a relational database but that does have some organizational properties that make it easier to analyze. Examples of semi-structured data might include XML, JSON documents and NoSQL databases.

## 2. Web aplikacija i relacione baze podataka. Prednosti korišćenja I nedostaci.

### Prednosti korišćenja RDBMS:

- Efikasno skladištenje podataka
- Podrška za ACID transakcije
- Podrška za kompleksne SQL upite
- Ogromna tehnološka baza (različiti DBMS-ovi, alati, programski interfejsi i sl.)

Za određene tipove problema, relacione baze jednostavno nemaju rešenje (društvene mreže i semantičke veze).

Web aplikacije imaju drugačije potrebe u odnosu na aplikacije za koje su RDBMS razvijane.

### Web aplikacije zahtevaju:

- Ekstremno veliki broj transakcija u jedinici vremena
- Dinamička analiza velikih količina podataka
- Kratko i predvidivo vreme odziva (latency)
- Visok nivo dostupnosti (high availability)
- Fleksibilnu šemu / polustruktuirane podatke
- Skalabilnost (po niskoj ceni)
- Geografska distribuiranost (veći broj čvorova u kojima se podaci obrađuju, mreža kao problem)

### Web aplikacijama nisu neophodne:

- Transakcije ??????
- Kompleksni SQL upiti
- Stroga konzistentost
- Integritet podataka

### Nedostaci RDBMS

- ACID transakcije nisu skalabilne (Horizontalno particionisanje, Neefikasni spojevi)

- Transakcije zahtevaju nepotrebnu obradu, odnosno unose dodatni “overhead”
- Šema relacionih baza podataka nije fleksibilna

### 3. Distribuirane baze podataka: pojam, skalabilnost i zahtevi koje moraju da zadovolje.

Distribuirane baze podataka - Više čvorova, jedna baza podataka.

Scaling Up - Dodavanje resursa jedinom čvoru u sistemu (dodavanje CPU ili memorije). Migracija sistema na jaču platformu. Prednosti: Brzo i jednostavno. Nedostaci: Kada se prevaziđu kapaciteti najačeg sistema, Cena, Zavisnost od samo jednog proizvođača

Scaling Out - Dodavanje novih čvorova u sistem. Prednosti: fleksibilnost. Nedostaci: kompleksnost.

- Funkcionalna (vertikalna) skalabilnost - Grupisanje podataka po funkciji i distribuiranje funkcionalnih grupa u različitim bazama
- Horizontalna skalabilnost - Distribuiranje istih funkcionalnih grupa u različitim bazama.

Zahtevi koje moraju da ispune distribuirane baze podataka:

- Consistency – sistem se nalazi u konzistentnom stanju posle svake operacije. Svi klijenti vide iste podatke
- Availability – sistem je uvek dostupan (“always on”) “no downtime”. Tolerancija na otkaz čvorova – klijenti uvek imaju pristup nekoj od kopija (replika). Tolerancija na HW/SW promene
- Partition tolerance – sistem funkcioniše čak i u slučaju da ne postoji konekcija između distribuiranih podskupova (pad mreže). Ne samo za čitanje već i za upis

### 4. CAP torema, CAE trade-off, BASE, poređenje BASE i ACID.

CAP Teorema - U potpunosti je moguće zadovoljiti samo 2 od 3 zahteva. Kompromis oko trećeg zahteva.

Biraju se različiti nivo konzistentnosti, dostupnosti ili particionisanja. Treba prepoznati koja su od CAP pravila neophodna za funkcionisanje sistema.

Kompromis oko Partition Tolerance

- Karakteristična za single-site cluster rešenja (lakše je obezbediti da su svi čvorovi u stalnom kontaktu)
- Kada dođe do narušavanja topologije mreže, odnosno do particionisanja mreže, sistem se blokira.

Kompromis za Availability

- Pristup pojedinim podacima može biti privremeno onemogućen ili ograničen
- Ostatak sistema se nalazi u konzistentom/tačnom stanju

Kompromis za Consistency

- Sistem je dostupan i prilikom narušavanja mrežne topologije
- Neki od podataka koje sistem vraća mogu biti privremeno neažurni (temporarily not up-to-date)

CAE trade-off (Amazon): ◦Cost-efficiency◦High Availability◦Elasticity

Biraju se bilo koja dva (C, A, E)

- Klijent čeka kada je sistem opterećen (C i E)
- Ukoliko je moguće predvideti opterećenje, moguće je obezbediti A i C rezervisanjem resursa unapred
- Nepotrebni resursi (over-provisioning) – A i E

BASE - CAP varijanta ACID svojstava ??????????

- Basically Available
- Soft State
- Eventually Consistent

ACID forsira konzistentnost podataka dok BASE prihvata da će se konflikti desiti.

### 5. Pojam NoSQL baza podataka, karakteristike, dobre i loše strane.

Termin se upotrebljava za sve nerelacione baze podataka (non-RDBMS)

- **Velike količine podataka** - Za skladištenje podataka se koristi distribuirana arhitektura
- **Veliki broj upita** - Nemogućnost efikasnog izvršavanja spojeva kod RDBMS u takvom okruženju
- **Schema evolution** - Nije jednostavno obezbediti fleksibilnost šeme. Promene u šemi se mogu postepeno uvoditi kod NoSQL.

**Dobre strane:** Fleksibilnost, Skalabilnost, Eventually consistent, Jeftine (osnova, infrastruktura je skupa), Prilagođene potrebama Web aplikacija

**Loše strane:** Tehnologija još uvek nije stabilna, Ne postoje zajednički standardi, Loša podrška za transakcije, Loša podrška za pretraživanje podataka, Zahteva promenu načina razmišljanja, Vrlo je teško naći dva identična scenarija primene.

## 6. Otklonomija NoSQL baza podataka.

**Key/Value stores** - Key/Value lookups (DHT), Hash. Jedna vrednost, jedan ključ, nema duplikata, izuzetno brzo. Skaliranje ogromnih količina podataka. Projektovane da podnesu velika opterećenja. Podataka je obično BLOB, DB ne razume strukturu podatka.

**Column stores** - BigTable kolonovi. Rasuta, distribuirana multi-dimenzionalna sortirana mapa. Konceptualno: Jedna tabela, beskonačno velika. Svaka vrsta može imati različite kolone (po broju i tipu). Tabela je retko posednuta:  $|rows| * |columns| > |values|$ .

**Document stores** - Key/Value store, value predstavlja polu-strukturirani dokument čija je struktura razumljiva DB. Podaci se mogu pretraživati ne samo po ključu

**Graph databases** - Inspirisane matematičkom teorijom grafova. Skalabilnost / kompleksnost podataka. Model: Key/Value parovi za Potege/Čvorove. Relacije: Potezi između čvorova

## 7. Osnovne karakteristike document baza podataka. Objasniti na primeru MongoDB baze podataka.

**Document store baze podataka** predstavljaju softver koji se koristi za snimanje, preuzimanje i modifikovanje polu-strukturiranih podataka, dokumenata. Document store baze su zapravo key/ value baze u kojima je value dokument.

Dokument store znaci da:

- **Objekti (podaci) se cuvaju kao dokumenti.** Tako da ne postoji object-relational impedance mismatch.
- **Dokumenti mogu da budu jako kompleksni.** Citav objektni model moze da se ucita odjednom, nema potrebe da za velikim brojem CRUD atomicnih operacija.
- **Dokumenti su nezavisni.** Povecavaju se preformansi i smanjuju problemi za konkurentan pristup podacima.
- **Otvorenog su formata.** Najcesce se koriste neki siroko korisceni formati kao sto su XML, JSON, BSON...
- **Ne postoji striktna sema cuvanja podataka.** Povecava se fleksibilnost seme. Tako se mogu dodavati novi podaci novog formata.
- **Cuvanje starih verzija.** Neke baze podataka cuvaju informacije o prethodnim verzijama dokumenta u toj bazi.

Vecina ovih baza obezbedjuje **RESTful api**, kao i **http servere** koji podrzavaju standardne http zahteve (put, get, post, delete).

**Pretrazivanje:** Osim pristupa dokumentima pomocu kljucua, one sadrze API pomocu kog je moguće detaljnije pretrazivati pazu po nekim atributima objekata. Ne postoji standardni jezik za pretragu.

**Za organizaciju** dokumenata koristi se:

1. Kolekcije
2. Mehanizam tagova
3. Hierarhije direktorijuma
4. Metapodaci

Jedna od najosnovnijih karakteristika dokument stor baza podataka je **eventual consistency**. Tako da se povremeni javlja nekoegzistentost podataka. Na racun toga ove baze podataka imaju povecanu skalabilnost i pristup podacima.

**Koriste se za:**

- **Dinamički podaci** - CMS (Content Management Systems) i CRM (Customer Relationship Management) objekti koje korisnici mogu da menjaju i prilagođavaju sopstvenim potrebama.
- **Polustrukturirani podaci**
- **Web podaci** – sesije, logovi, shoppin cart i drugi podaci koji se održavaju za potrebe Web aplikacija. Document stores omogućavaju da se podacima pristupa kao celini jednim zahtevom ka udaljenom serveru.
- **Obrada velike količine podataka** – dobra skalabilnost document store rešenja i podrška za distribuiranu obradu.

**Problemi:**

- Kada je potrebno koristiti transakcije (nema podrške za ACID svojstva)
- Kada je potrebno koristiti SQL upite.

Neki pretstavnici dokument store baza podataka su: **MongoDB**, CouchDB..

**MongoDB**

**Koristi** JSON za predstavljanje podataka. Interno se koristi BSON. Osnovni tipovi podataka: number, string, array, bulean, object, null.

**MongoDB indeksira** dokumente u strukturi B-stabla. Oni sluze za brze pribavljanje dokumenata, ali usporavaju operacije azuriranja. Oni mogu da se kreiraju nad poljem dokumenata, nad delom polja ili nad vise polja.

Za svaku kolekciju u MongoDB-u postoji **\_id** jedinstveni identifikator kolekcije. Za njega se najcesce koristi **Object ID** vrednost.

**Sekundarni indksi** – njih moze da kreira korisnik, kao i da odredi nacin sortiranja polja koja su indeksirana. Oni se prave pomocu operacije **ensureIndex()**.

**Ugnjezdeni indksi** – Moguce je koristiti prefiksnu notaciju za predragu po njima.

**Unique index** – sprecava dupliranje vrednosti u poljima koji su indeksirana.

**Sparse indksi** – dozvoljava indeksiranje dokumenata samo ukoliko imaju definisanu vrednost indeksiranog polja.

**Ogranicen** je na velicinu dokumenta od 16MB. Za pamcenje vecih dokumenata koristi se GrigFS. Za to se koriste 2 vrste kolekcija: files i chunks

**Replikacija** – postoje primarne i sekundarne replikacije. Upisivanje se vrsi u primarne instance, a nakon toga se sekundarne asinhrono updejtuju. Ukoliko se primarna kopija izgubi, ostale masine glasaju na kojoj masini ce da se nadje nova primarna instanca.

**Sharding mehanizam** – on služi za partitionisanje kolekcije i njihovu distribuciju na više instanci. Partitionisanje se vrši na osnovu ključa dokumenata. Kada particija preraste kapacitet instance ona se deli i raspoređuje na više instanci. Automatski se balansira količina podataka na različitim instancama. Ovaj proces distribuiranja podataka je potpuno transparentan.

**Map/Reduce** – Model za distribuciju obrade velike količine podataka. Odvija se u 2 koraka:

1. Map – master cvor deli zahtev na sitnije zahteve i daje ih vorker cvorovima na obradu
2. Reduce – master cvor prikuplja i spaja podatke od vorker cvorova.

## 8. i 9. Osnovne karakteristike Column-store baze podataka. Objasniti na primeru Google BigTable / Cassandra modela podataka.

### Column-store baze podataka

Predstavljaju klonove BigTable-a.

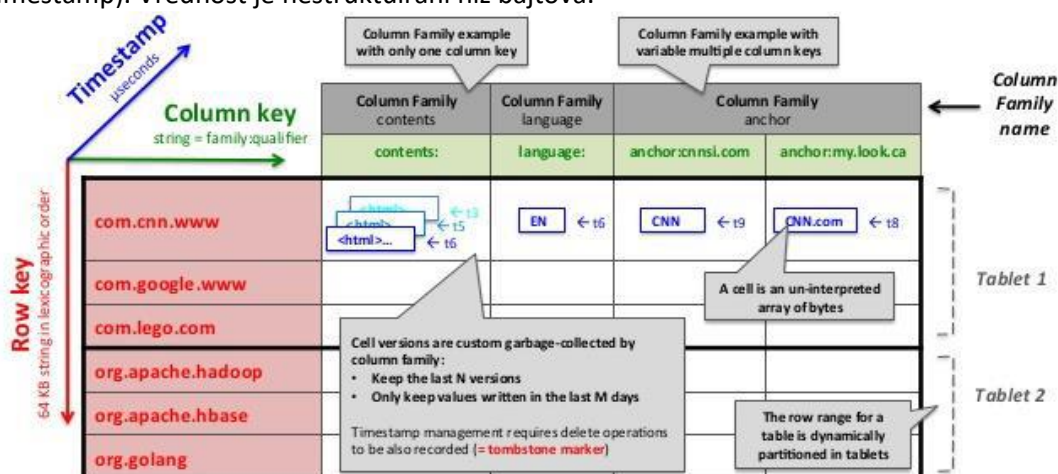
To je u stvari rasuta, distribuirana multi dimenzionalna sortirana mapa.

Konceptualno:

- Jedna beskonacno velika tabela.
- Svaka vrsta može imati različite kolone (po broju i sadržaju)
- Retko posednuta tabela  $|vrste| * |kolone| > |vrednosti|$

### BigTable

BigTable - Rasuta, distribirana, perzistentna multi-dimenzionalan sortirana mapa. Dimenzije (row, column, timestamp). Vrednost je nestruktuirani niz bajtova.



**Vrste:** Ključevi vrsta su proizvoljni stringovi. Podaci su leksikografski sortirani po ključu vrste. Leksikografski bliske vrste se obično nalaze na istom server ili na malom broju server. Pristup kolonama u vrsti je atomičan.

**Kolone, Familije kolona:** Ključevi kolona su proizvoljni stringovi. Neograničeni broj kolona. Familiji kolona se pridružuje informacija o tipu podataka.

**Timestamp:** Svaka ćelija može da ima više vrednosti. **Verzije** (svaki podatak u bazi može imati više vrednosti u zavisnosti od verzija koje se čuvaju):

- Automatski garbage collection
- Čuva se N poslednjih verzija
- Čuvaju se samo verzije novije od zadatog timestamp-a

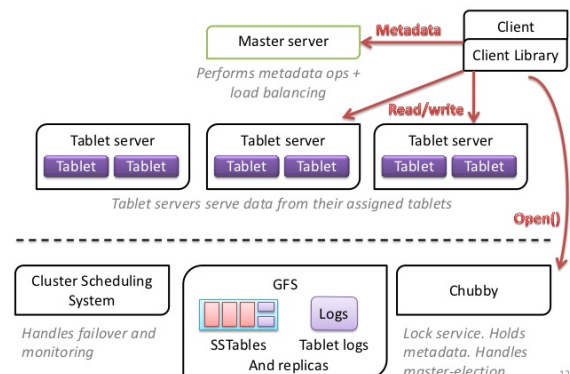
**Tableti:** Opseg vrsta je dinamički partitionisan u tablete (sekvence vrsta). Range scan operacije su jako efikasne. Ključevi vrsta se biraju tako da povećaju lokalnost operacija pristupa podacima (bliski podaci treba da budu u istom tabletu)

#### Arhitektura:

- Podaci se čuvaju u Google file system-u (GFS)
- Chubby (distribuirani lock servis)
- Map/Reduce
- 1 master server/ mnogo tablet servera

**Master server:** Operacije sa metapodacima. Balansira opterećenje tablet servera. Garbage collection. Upravljanje šemom. Klijent ne pristupa master serveru već direktno tablet serverima. Master server ne upravlja lokacijom tableta

### Bigtable System Architecture

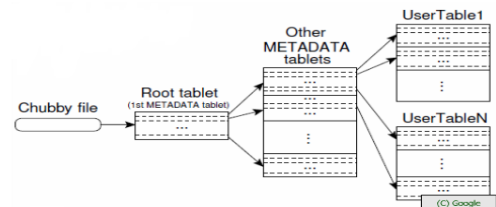


**Tablet server:** Veliki broj tablet servera. Upravlja Read/Write/Split operacijama nad tabletima.

#### Lokacija tableta:

1. Root (Chubby file) –lokacija Root Metadata tableta.
2. Prvi nivo (Root Metadata tablet) – lokacija svih metadata tableta.
3. Drugi nivo (Metadata tablet) – lokacija skupa tableta sa podacima.

Struktura B+stabla. Klijenti keširaju lokacije podataka.



#### Ažuriranje:

- Ažurirase redo-log
- Ažurirase in-memory buffer (memtable)
- Ukoliko je memtable buffer pun, kreirase nova SStable i memtable se prazni.

(U log se upisuje šta se menja. Ažurira se ili se upisuje podatak u in-memory buffer. Kad se taj bafer napuni, dodaš u bazu nov SStable. Ovo se radi jer se podaci nikad ne brišu, samo se dodaju nove verzije, tako da ažuriranje podataka koji nisu u in-memory u stvari nije ažuriranje, nego dodavanje novih verzija.)

#### Čitanje:

- Izvršava se nad memtable i svim SStable
- Pristup memtable je izuzetno efikasan (podaci su unapred sortirani)
- Koriste se **Bloom filteri** –mape koje definišu da li određena SStable sadrži odgovarajuću ćeliju (vrsta, kolona) -drastično smanjen broj pristupa fajl sistemu

### Cassandra

**Cassandra** - Open source distribuirani sistem za upravljanje bazom podataka. Podržava upravljanje velikom količinom distribuiranih podataka bez postojanja single point of failure. Bazirana na Google BigTable i Amazon Dynamo. Prenosi: skalabilnost, dostupnost, performanse.

Cassandra ispunjava A i P iz CAP teoreme. C može da se podešava (Eventual consistency).

**Osnovu modela podataka** čini sortirana hash mapa. Mapa obezbeđuje efikasne operacije pretage podataka po ključu. Sortiranost obezbeđuje efikasno skeniranje opsega. Neograničeni broj kolona. Ključ može da predstavlja podataka sam po sebi. (Generalno arhitektura BigTable)

**Keyspace** odgovara konceptu relacione baze podataka. Ima attribute:

- Faktor replikacije (koliko replika ima instanca podataka)
- Faktor distribucije (kako se distribuira)
- Koje column family sadrži

**Column family** odgovara konceptu tabele kod relacionih baza podataka. Predstavlja kontejner za kolekciju vrsta. Svaki podatak se može tretirati kao četvorodimenzionalna hash mapa: [Keyspace][Column family][Key][Column]. Column family sadrži kolone ili super kolone (kolona koja se sastoji od podkolona). Svaka columnfamily se skladišti u zasebnoj datoteci na disku pa je preporučljivo da se povezane kolone skladište u okviru iste columnfamily.

#### Tipovi vrsta:

- Wide rows –veliki broj različitih kolona i mali broj vrsta (koriste se za čuvanje lista objekata)
- Skinny rows –mali broj kolona, veliki broj vrsta (bliže relacionom modelu podataka)

**Superkolona** predstavlja mapu podkolona. Superkolona ne može da sadrži druge superkolone. Petodimenzionalna hash mapa: [Keyspace][Column family][Key][SuperColumn][Subcolumn]. U pojedinim situacijama je bolje koristiti kompozitne ključeve.

**Pripadnost klasteru: Gossip protokol** –svaki čvor komunicira sa 1-3 susedna čvora o stanju klastera (razmenjuje informacije). Promene u klasteru (dodavanje/uklanjanje čvorova, otkazi) se brzo propagiraju. Tehnike zasnivane na verovatnoći za otkrivanje otkaza.

**Dinamičko particionisanje:** Konzistentan hash mehanizam. Prsten čvorova. Čvorovi mogu da menjaju poziciju u prstenu zbog balansiranja opterećenja.

**Write** (gotovo== BigTable) Klijenti šalju zahteve random čvoru, čvor koji primi zahtev određuje čvor odgovoran za podatke. Podaci su replicirani u N čvorova. Podaci se prvobitno daju commit log, zatim u memtable i na kraju u SSTable.

**Reads** (gotovo== BigTable) Zahtev se šalje random čvoru, čvor koji primi zahtev ga prosleđuje ka N čvorova koji poseduju podatke. Varijante:

- SingleRead –podaci iz prvog čvora koji sadrži podatke
- QuorumRead –vrednost oko koje se slaže većina čvorova ( $N/2+1$ )
- FullRead –vrednost oko koje moraju da se slože svi čvorovi. Podaci se prvo čitaju iz mem table, nakontoga iz SSTable, Koriste se Bloom filteri.

Kompakcije(== BigTable)

## 10. Osnovne karakteristike graf baza podataka. Objasniti na primeru Neo4J baze podataka.

### Graf baze podataka

Graf baze podataka koriste strukturu podataka tipa GRAF (čvorovi, potezi i svojstva) za reprezentaciju i skladištenje podataka. Po definiciji, graf baza podataka je bilo koji sistem za skladištenje podataka koji omogućava definisanje relacija nezavisno od indeksnih i lookup struktura.

**Specijalizovane graf baze podataka:**

- **Triplestore** (skladištenje podataka u obliku subjekat-predikat-objekat, RDF)
- **Mrežne baze podataka** (baziraju se na korišćenju mrežnog modela)

**Elementi:** čvorovi, potezi i svojstva.

- **Koncept čvorova** je sličan konceptu objekta u OO programiranju.



- **Čvorovi** se koriste za reprezentaciju entiteta.
- **Svojstva** su podaci koji opisuju čvorove.
- **Potezi** su relacije koje povezuju čvorove međusobno ili čvorove sa svojstvima.
- **Svojstva** mogu biti pridružena i **potezima**.

Da bi se ubrzao pristup i pretraga čvorova mogu se **indeksirati pojedina svojstva**. Podaci iz graf baza podataka se direktno mapiraju na OOmodel. Graf baze podataka omogućavaju organizaciju podataka u različite kompleksne strukture povezanih podataka. One su dobro rešenje za velike količine podataka pošto ne zahtevaju skupe operacije spoja. Zahvaljujući fleksibilnoj šemi pogodno za razvoj ad-hoc aplikacija ili aplikacija koje brzo evaluiraju.

**Pogodne za operacije tipa:** obilazak grafa, traženje najkraćeg puta između dva čvora, proverda da li postoji put između dva čvora, detektovanje paterna, određivanje suseda i sl.

#### Neo4J

**Graf baza podataka + Lucene Index.**

**Property graf.**

**FullACID** (atomicity, consistency, isolation, durability) – garantuje konzistentnost podataka za sve operacije koje se izvršavaju unutar transakcija.

**Relacije** između čvorova predstavljaju ključni element grafa. Omogućavaju pronalaženje povezanih podataka.

**Svojstva** predstavljaju key-value parove. Vrednosti svojstava mogu biti atomične vrednosti ili nizovi atomičnih vrednosti.

**Putanja** (path) predstavlja kolekciju (najmanje jedan) čvorova koji su međusobno povezani relacijama. Putanja se dobija kao rezultat upita odnosno rezultat obilaska grafa.

**Cypher** –deklarativni upitni jezik za grafove. Omogućava CRUD operacije nad Neo4J grafovima. Izgrađen je na elementima SQL-a

## 11. i 12. Osnovne karakteristike key-value baza podataka. Objasniti na primeru Redis baze podataka/ Dynamo modela podataka.

#### Key/Value

- Key/Value paradigma za pristup podacima obezbeđuje dobre performanse i dostupnost podataka. Jedna vrednost, jedan ključ, nema duplikata, izuzetno brzo.
- Skaliranje ogromnih količina podataka.
- Projektovane da podnesu velika opterećenja.
- Ključevi i vrednosti mogu biti kompleksni objekti.
- Konzistentnost podataka se može primeniti samo na operacije koje se odnose na jedan ključ.

Key/Value lookups (Dictionary Hash Table) - Klasična Hash tabela. Paradigma za skladištenje, pribavljanje i upravljanje podacima koji su smešteni u obliku asocijativnih nizova (associative array). Dictionary predstavlja kolekciju objekata (objects/records) pri čemu svaki objekat može imati svoj kolekciju atributa i njihovih vrednosti. Podatak može biti predstavljen kao BLOB objekat pri čemu baza podataka ne razume strukturu podatka. Svaki objekat je jedinstveno identifikovan korišćenjem ključa.

#### Prednosti:

- Efikasne pretrage podataka po ključu (predvidive performanse).
- Mogućnost jednostavne distribucije podataka u cluster-u.
- Fleksibilnost šeme – struktura podataka se može razlikovati od ključa do ključa.
- Memorijska efikasnost – predstavljaju se samo atributi čije vrednosti postoje (kod relacionih baza podataka svaki podatak mora da ima sve attribute).
- Ne postoji problem object-relational impedance mismatch.

- Korišćenje relacione baze podataka u sprezi sa sistemom zakeširanja forsira korišćenje key/value arhitekture.

#### **Nedostaci:**

- Nepostojanje kompleksnih upita (kompleksnih filtara nadpodacima)
- Nepostojanje stranih ključeva
- Za kreiranje spojeva između različitih podataka odgovorna je aplikativna logika.
- Nedostatak standardizacije

#### **Redis**

**Redis** predstavlja najpopularniju implementaciju key/value paradigme.

U osnovi Redis predstavlja in-memory dataset. Perzistencija podataka može da se ostvari na dva načina:

- **Snapshooting** – kompletan skup podataka iz memorije se periodično snima na disk
- **Append-only File (AOF)** – nakon izmena podataka u memoriji, te izmene se upisuju u append-only datoteku na disku (journal)

Podrazumevano sinhronizacija podataka između memorije i diska se vrši na dve sekunde. Po potrebi frekvencija može da se poveća ili smanji. U slučaju kompletnog otkaza biće izgubljene sve izmene koje su se desile nakon poslednje sinhronizacije.

Ne postoji značajna razlika u performansama između operacija čitanja i pisanja. Redis server je implementiran kao jedan proces sa jednom niti. Ne postoji mogućnost paralelizacije operacija kod jedne instance Redis servera.

Za **ključ** se mogu iskoristiti bilo koje binarne sekvence (binary safe je). Treba izbegavati isuviše duge ključeve koji mogu da dovedu dopada performansi zbog operacija poređenja prilikom pretrage podataka. Treba izbegavati isuviše kratke ključeve kako bi se povećala čitljivost podataka. Redis omogućava definisanje ključeva sa ograničenim vekom trajanja.

Redis podržava skladištenje podataka predstavljenih uvidu različitih **struktura podataka**:

- Stringovi
- Liste
- Skupovi (Sets)
- Sortirani skupovi (Sorted sets)
- Bitmape (Bit arrays)
- HyperLogLogs

U većini slučajeva Redis je single-threaded i sve operacije su: **Atomične, Konzistentne, Izolovane**.

**Trajnost (durability)** je konfigurabilna i predstavlja kompromis između efikasnosti i sigurnosti podataka.

**Redis transakcija** omogućava izvršavanje skupa komandi u vidu jedne celine. Redis transakcija obezbeđuje da se tokom izvršavanja komandi jedne transakcije ne mogu izvršavati komande izdate od stranenelog drugog klijenta.

Redis podržava master/slave **mehanizam replikacije**. Podaci sa jednog Redis servera mogu se replikovati na veći broj slave servera. Slave server može postati master za neki drugi Redis server. Time se može implementirati stablo replikacije sa originalnim master serverom u korenu stabla (single-rooted replication tree). Redis slave server može da prihvata i write operacije čime se dozvoljava postojanje nekonzistentnosti između čvorova u stablu. Replikacije je pogodna u slučaju kada je potrebno obezbediti skalabilnost read operacija (ali ne i write operacija) ili redundantnost podataka.

Redis Cluster obezbeđuje automatsko particionisanje podataka između većeg broja instanci Redis servera.

## Dynamo

Dynamo is the name given to a set of techniques that when taken together can form a highly available key-value structured storage system or a distributed data store. It has properties of both databases and distributed hash tables (DHTs). Amazon DynamoDB is "built on the principles of Dynamo". It was created to help address some scalability issues.

### Principles

- **Incremental scalability:** Dynamo should be able to scale out one storage host (henceforth, referred to as "node") at a time, with minimal impact on both operators of the system and the system itself.
- **Symmetry:** Every node in Dynamo should have the same set of responsibilities as its peers; there should be no distinguished node or nodes that take special roles or extra set of responsibilities.
- **Decentralization:** An extension of symmetry, the design should favor decentralized peer-to-peer techniques over centralized control.
- **Heterogeneity:** The system needs to be able to exploit heterogeneity in the infrastructure it runs on. e.g. the work distribution must be proportional to the capabilities of the individual servers. This is essential in adding new nodes with higher capacity without having to upgrade all hosts at once.

## 13. Objasniti pojam BigData.

Big data is a term for data sets that are so large or complex that traditional data processing application softwares are inadequate to deal with them. Challenges include capture, storage, analysis, data curation, search, sharing, transfer, visualization, querying, updating and information privacy. The term "big data" often refers simply to the use of predictive analytics, user behavior analytics, or certain other advanced data analytics methods that extract value from data, and seldom to a particular size of data set. Data sets grow rapidly - in part because they are increasingly gathered by cheap and numerous information-sensing mobile devices, aerial (remote sensing), software logs, cameras, microphones, radio-frequency identification (RFID) readers and wireless sensor networks. The world's technological per-capita capacity to store information has roughly doubled every 40 months since the 1980s; as of 2012, every day 2.5 exabytes ( $2.5 \times 10^{18}$ ) of data are generated.

Relational database management systems and desktop statistics - and visualization-packages often have difficulty handling big data. The work may require "massively parallel software running on tens, hundreds, or even thousands of servers". Big data usually includes data sets with sizes beyond the ability of commonly used software tools to capture, curate, manage, and process data within a tolerable elapsed time. Big data "size" is a constantly moving target. Big data requires a set of techniques and technologies with new forms of integration to reveal insights from datasets that are diverse, complex, and of a massive scale.

META Group (now Gartner) defined data growth challenges and opportunities as being three-dimensional, i.e. increasing volume (amount of data), velocity (speed of data in and out), and variety (range of data types and sources). "Big Data represents the Information assets characterized by such a High Volume, Velocity and Variety to require specific Technology and Analytical Methods for its transformation into Value".

## 14. HADOOP – pojam i osnovne karakteristike.

HADOOP je platforma za skladištenje i obradu velike količine podataka na velikom broju distribuiranih mašina. Do 4000 računara u klasteru i 20PB podataka.

### Osnovno:

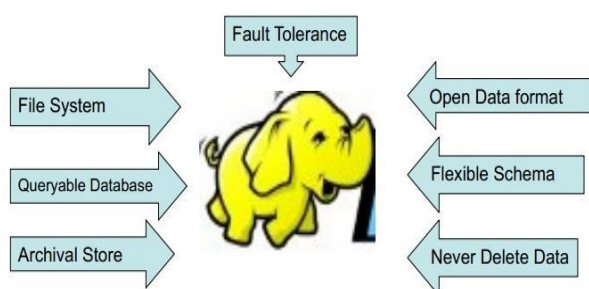
- Bazira se na korišćenju infrastrukture jeftinih i dostupnih računara. Ne zahteva superračunare.
- High reliability ( Automatski fail-over za podatke i obradu).
- Obrada podataka koja se meri u PT.

Jako je skupo obezbediti pouzdanost za svaku zasebnu aplikaciju, zato je HADOOP projektovan sa idejom da će se **otkazi dešavati**:

- Otkaz čvora nije izuzetak, već je uobičajena pojava
- Broj čvorova u cluster-u nije konstantan

**Razlika** između **hadupa** i **relacione baze** podataka:

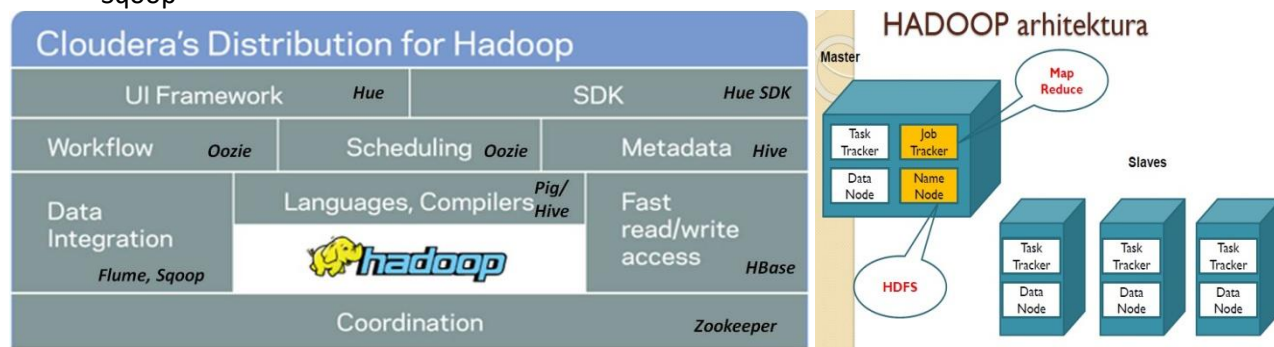
- Brz upis podataka vs Brzo citanje podataka
- Scale out vs scale up
- Key/value vs table
- Definise se kako obradivati podatke vs definišu se potrebni podaci
- Fleksibilnost/ agilnost vs standardi/ upravljanje



## 15. HADOOP ekosistem – osnovni elementi, arhitektura i gde se primenjuje.

**HADOOP ekosistem** se sastoji od kolekcije softverskih komponenti:

- HADOOP Common
- Hadoop Distributed File System (HDFS)
- MapReduce
- HBASE
- HIVE
- Pig
- Sqoop



Primena: Search, Log processing, Clickstream, Recommendation Systems, Sentiment analysis, Data Warehouse, Video and Image Analysis

## 16. HDFS – osnovni pojmovi.

HADOOP Distributed File System - Distribuirani, skalabilni, portabilni file system razvijen za potrebe HADOOP ekosistema. Podrazumevano skladište podataka za HADOOP cluster. Virtuelna arhitektura. Konfigurabilna replikacija.

- **Namespace** - Metapodaci o datotekama. Razdvojeni od podataka. Brze operacije.
- **NameNode** - Kompletan Namespace se čuva u memoriji
- **DataNode** - Podaci se čuvaju u vidu distribuiranih blokova

**Replikacija:** Datoteka se deli na blokove, a zatim se blokovi repliciraju unutar klastera (default=3 kopije, 1 kopija u drugom rack-u).

#### Replikacija blokova:

- Trajnost/otpornost na otkaze
- Dostupnost
- Propusna moć

## 17. Pojam MapReduce tehnike.

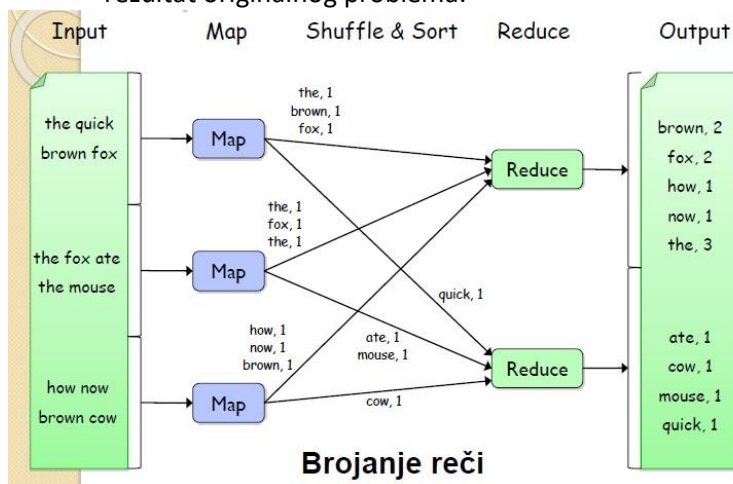
Okruženje za razvoj distribuiranih aplikacija za obradu podataka. MapReduce job se odvija u dve faze:

- **Map** (Kin,Vin) -> list(Kinter,Vinter)– distribuirana faza
- **Reduce** (Kinter, list(Vinter)) ->list(Kout,Vout)– faza koja ne mora da bude distribuirana

Combiners i Partitioners – pomoćne faze koja imaju zadatak da optimizuju obradu podataka.

**Map/Reduce** - Tipična namena je distribucija obrade na veći broj čvorova. Omogućava paralelnu obradu velike količine podataka. Odvija se u dva koraka:

1. **Map** – master čvor prihvata ulaz, deli ga na manje potprobleme i distribuira ih worker čvorovima. Worker čvor može ponoviti isiti korak kreirajući stablo čvorova koji obavljaju procesiranje podataka. Worker čvor obrađuje manji problem, i rezultate obrade prosleđuje master čvoru.
2. **Reduce** – master čvor prikuplja od worker čvorova rezultate obrade potproblema i kombinuje ih u rezultat originalnog problema.



#### Otpornost na otkaze:

##### 1. Otkaz task-a

- Task se ponovo pokreće na drugom čvoru
- Map i Reduce su nezavisne faze
- U slučaju da se otkazi ponavljaju, poništava se job ili se zanemaruju podaci

##### 2. Otkaz čvora

- Svi taskovi se ponovo pokreću na drugom čvoru
- Pokreću se svi map taskovi (otkazom čvora nestaju međurezultati)

##### 3. Sporo izvršavanje task-a

- Kopija taska se pokreće na drugom čvoru
- Prihvata se rezultat taska koji se prvi završi

## 18. HBASE – osnovne karakteristike, model podataka, primeri korišćenja, dobre i loše strane korišćenja.

**HBASE**- Distribuirana column-oriented baza podataka implementirana korišćenjem HADOOP ekosistema.

Koristi se u situacijama kada je neophodno obezbediti realtime read/write random-access za velike skupove podataka.

**Obezbeđuje:** row-level ažuriranja, kratko vreme odziva za upite, transakcije.

**Tabele:** velike, retko popunjene podacima, slabo strukturane

- **Sastoji se** od vrsta, jedinstveni ključevi vrsta
- **Vrste** imaju različiti broj kolona
- **Kolone** su grupisane u familije
- **Dinamičko kreiranje kolona**

**Model podatka:** Row key – Column family – Value - Timestamp

**Kada koristiti HBASE:**

1. **Nije pogodan sve probleme** - Ograničen API u odnosu na RDBMS
2. **Za velike količine podataka** - Veliki broj vrsta. Ukoliko je količina podataka mala, sve vrste će biti raspoređene u jednom čvoru a ostatak sistema će biti neiskorišćen.
3. **Odgovarajući hardver je na raspolaganju** - Veliki broj procesa. Zahtevan sa stanovišta memorija i procesora. Minimalno 5 čvorova.
4. Dobre performanse kod sistema **sa velikim brojem zahteva**
5. Pogodan za scenarija koja zahtevaju **veliki broj random pristupa ili range scan pristup**
6. **Fleksibilna šema** - Različite vrste imaju različite kolone. Veliki broj kolona pri čemu većina od njih nema definisanu vrednost.

	RDBMS	HBase
Data layout	Row-oriented	Column-family-oriented
Transactions	Multi-row ACID	Single row only
Query language	SQL	get/put/scan/etc *
Security	Authentication/Authorization	Work in progress
Indexes	On arbitrary columns	Row-key only
Max data size	TBs	~1PB
Read/write throughput limits	1000s queries/second	Millions of queries/second

## 19. Vremenske serije podataka

Promenljiva čije se vrednosti mere u određenim tačkama u vremenu se naziva **vremenskom serijom podataka (time series)**.

Vremenski intervali između tačaka su često jednaki.

**Namena:**

- **Razumevanje** pojava i struktura koje su proizvele određenu vremensku seriju podataka.
- **Nadgledanje (monitoring)** tih istih pojava i struktura.
- **Predviđanje** budućih vrednosti.

Postoji četiri obrasca ili komponenti kojima se mogu opisivati promene vrednosti vremenske serije podataka, a to su:

1. **Opšti trend** - Opštim trendovima se nazivaju dugoročne promene vrednosti vremenske serije podataka. Npr., socioekonomski faktori kao što su zaposlenost, visina plata itd. se obično menjaju u trendovima.
2. **Sezonska komponenta** - Sezonska komponenta predstavlja promene u nešto kraćim periodima koje se dešavaju usled određenih sezonskih uslova. Sezona, u ovom smislu, može biti: godišnji kvartal, mesec, nedelja itd.
3. **Ciklična komponenta** - Ciklične komponente imaju podaci koji pokazuju rastove ili padove koji nisu u periodima fiksne dužine i po tome se razlikuju od sezonskih trendova. Da bi mogle biti proučavane, ovakve vremenske serije podataka obično moraju posedovati što više tačaka, sa što manje slučajnih komponenti.
4. **Slučajna komponenta** - Ovakve varijacije se nekada mogu nazivati iregularnim fluktuacijama. Te promene, iako su slučajne u prirodi, mogu nagovestiti kontinualnu promenu u trendovima u predstojećem periodu.

## 20. Objasniti pojam time series baza podata

**Time series databases (TSDB)** - baze podataka za rad sa vremenskim serijama podataka.

Neke koriste **NoSQL pristup**, dok se druge baziraju na korišćenju tehnologije relacionih **baza podataka** u pozadini.

**Fundamentalni zahtevi** koje baza podataka rad sa vremenskim serijama podataka mora da zadovolji:

- **Ko-lokacija podataka**- Sistem koji smešta sortirane podatke u sekvencijalnom redosledu (ko-lokacija) na istim mestim.
- **Efikan pristup podacima u proizvoljnom interval**

**Osnovne karakteristike:**

1. **Performanse i skalabilnost** - Dobro projektovana TSDB omogućava lako skaliranje sistema tako da on može da podrži veliki broj informacija koje može da skladišti
2. **Kompakcija podataka (data compaction)**- Kako podaci neke vremenske serije podataka stare, to su manje bitne individualne tačke. Gotovo svaka TSDB ima određene mehanizme za degradiranje i združavanje zodataka.
3. **Niži troškovi** - Zbog optimizovanosti za rad sa vremenskim serijama, TSDB direktno utiču na smanjenje troškova zato što troše manje računarskih resursa pri radu sa vremenskim serijama podataka.
4. **Bolje poslovne odluke** - Time što se podaci mogu analizirati u realnom vremenu, organizacije mogu da prave bolje i preciznije odluke, kao i da detektuju i reaguju na potencijalne probleme na vreme.

## 21. Primeri time series baza podataka (focus na InfluxDB bazi podataka).

**Primeri**

- **RRDTool** - RRDtool je napisan u programskom jeziku C. Podaci se smeštaju u bazi koja je implementirana kao kružni bafer.
- **OpenTSDB**- Za skladištenje podataka koristi Apache HBase bazu podataka ili Google-ovim Bigtable servis.
- **KairosDB** - nastao kao fork OpenTSDB projekta. Za skladištenje podataka koristi Apache Cassandra ili H2 baze podataka.
- **RiakTS** – distribuirana NoSQL key-value baza podataka optimizovana za skladištenje vremenskih serija podataka. Predstavlja nadogradnju postojećeg Riak key-value rešenja I dele isti kod.



## InfluxDB

Zbog prirode vremenskih serija podataka koje se gotovo nikad ne ažuriraju (update) i ne brišu (delete), InfluxDB je dosta optimizovaniji za čitanje i pisanje.

### Arhitektura:

- **Telegraf** - Agent koji je zadužen za prikupljanje podataka iz različitih izvora (plug-in arhitektura).
- **Kapacitor** - Okruženje za obradu podataka (u realnom vremenu ili batch obradu).
- **Chronograph** - Korisnički interfejs za administraciju i vizelizaciju platforme.
- **InfluxDB** - Skladište za čuvanje vremenskih serija podataka

### Model podataka:

- **Vreme** (time) je tip kolone i interno se pamti kao UNIX timestamp.
- **Polja** (fields) se uglavnom koriste za smeštanje samih numeričkih vrednosti neke promenljive.
- **Tagovi** (tags) predstavljaju tip kolone koji se uglavnom koristi za neke propratne, meta-informacije koje mogu biti od značaja uz izmerene vrednosti vremenske serije podataka.

**Mera** (measurement) ili metrika se sastoji od minimum jedne kolone tipa vreme i jedne kolone tipa polje i kao takva, čini osnovnu strukturu podataka u InfluxDB-u. Mera može imati neograničen broj polja i tagova.

**InfluxQL** je naziv upitnog jezika koji je napravljen za potrebe InfluxDB-a i koristi se za pisanje upita

**Kontinualni upiti** su InfluxQL upiti koji se periodično izvršavaju nad podacima koji su dobijeni u realnom vremenu.

## 22. Objasnite pojam pretraživanja informacija

**Pretraživanje informacija** predstavlja aktivnost pribavljanje relevantnih resursa iz kolekcije resursa na osnovu zadatog kriterijuma. Resurs najčešće predstavlja nestrukturirani document.

**Full-text pretraga** predstavlja tehniku pretrage dokumenata kod koje se pretražuje ceo dokument tj. njegov sadržaj.

**Najbitniji aspekti** full-text pretrage:

- **Relevantnost** – mogućnost da se rezultati pretrage rangiraju po tome koliko su relevantni za dati upit (query).
- **Analiza** – mogućnost pretvaranja bloka teksta u posebne normalizovane tokene, na osnovu kojih se kreira indeks za pretragu.

**Osnovne aktivnosti** koje treba obezbediti kod svakog mehanizma za full-text pretragu:

- **Procesiranje i indeksiranje** dokumenata – obavlja se nad dokumentima tokom njihovog dodavanja u sistem. Dokumenti se svode na predefinisani standardni format za skladištenje.
- **Procesiranje upita** - interpretacija korisnikovog zahteva odnosno pitanja na osnovu čega se izvršava upit (query).

## 23. Objasnite pojam invertovanog indeksa.

**Standardni indeks** kod relacionih baza podataka.

**Karakteristike:**

- B/stablo
- Sortirano za range upite
- Složenost operacija  $O(\log(n))$

**Centralna struktura** koja se koristi kod full-text pretrage. Ovakva struktura omogućava izuzetno brzu pretragu, pri čemu zahteva dodatno procesiranje prilikom dodavanja novih dokumenata u kolekciju.



### Struktura invertovanog indeksa:

- **Vokabular** (vocabulary) – lista svih reči koje se pojavljuju u dokumentu ili skupu dokumenata.
- **Pojavljivanja** (Occurrences) - lista koja sadrži informacije o svakoj reči iz vokabulara (dokument u kojima se reč pojavljuje, pozicija u okviru dokumenta, frekvencija pojavljivanja i sl.).

## 24. Lucene biblioteke – namena i osnovne karakteristike.

**Lucene** je biblioteka za pretraživanje informacija koja omogućava skalabilnu pretragu sa visokim performansama.

☐ **Koristi se** za pretraživanje dokumenata, tj. Informacija unutar dokumenata ili metapodataka o dokumentima.

☐ **Bazira se** na implementaciji invertovanog indeksa.

**U teoriji** pretraživanja informacija najčešće se pominju tri modela pretraživanja:

- **Bulov model** – Dokumenti zadovoljavaju ili ne zadovoljavaju dati upit.
- **Vektorski model** – Dokumenti i upiti se modeluju kao vektori u višedimenzionalnom prostoru, a relevantnost se računa kao rastojanje između tih vektora.
- **Probabilistički model** – U ovom modelu se računa verovatnoća da dokument zadovoljava upit (koristi se u biblioteci Xapian).

Lucene biblioteka kombinuje vektorski model i Bulov model.

## 25. Elasticsearch – namena i osnovne karakteristike

NoSQL document store platforma skladištenje i pretragu dokumenata bazirana na korišćenju Lucene indeksa. JSON kao standardni format za skladištenje dokumenata.

☐ Pouzdano i skalabilno rešenje,

☐ Otpornost na pojavu grešaka.

☐ Podržane replikacije i distribuirano indeksiranje.

☐ Podržava balansiranje opterećenja prilikom izvršavanja upita.

## 26. Objasniti pojam semantičkog Web-a.

Semantički web je unapređenje postojećeg weba kod kojeg će informacijama biti jasno definisano značenje, na taj način omogućavajući računarima i ljudima bolju komunikaciju.

Web koji trenutno postoji je mesto gde: Računari obavljaju prezentaciju (što je lako), a ljudi obavljaju povezivanje i interpretaciju (što je teško). Kako bi računari mogli da obavljaju i teži deo posla potrebno je da razumeju informacije koje nam prikazuju. Ideja o postojanju podataka na Web-u koji su definisani i povezani tako da ih mogu koristiti mašine (računari, aplikacije) ne samo za prikaz već i za deljenje, automatizaciju i integraciju između različitih aplikacija. Metapodaci obezbeđuju mehanizme za definisanje značenja podataka.

Podaci mogu biti povezani na sličan način kao web strane. Kombinovanje podataka iz različitih izvora u cilju kreiranja novih činjenica. Mašine (agenti) mogu da koriste tehnike zaključivanja za izvođenje novih činjenica koje nisu eksplicitno zapamćene.

## 27. Objasniti pojam ontologija.

Ontologija predstavlja eksplicitnu specifikaciju konceptualizacije.

Danas imaju veliku primenu u oblasti računarstva za opisivanje znanja. Računarski čitljive definicije osnovnih koncepta u domenu i veza između njih.

Ontologija se sastoji od niza aksioma koji nameću ograničenja nad klasama, svojstvima, instancama i relacijama između njih. Ovi aksiomi omogućavaju izvođenje novih činjenica na osnovu eksplicitno zadatih podataka.

Ontologija se sastoji od određenog rečnika (vokabulara) koji se koristi za opisivanje određenog domena i skupa eksplicitnih pretpostavki u pogledu nameravanog značenja rečnika. Ontologija definiše formalnu specifikaciju određenog domena.

Resource Description Framework (RDF)

Namena: °Definisanje podataka i metapodataka °Definisanje struktura skupova podataka °Definisanje relacija između delova podataka

RDF Schema (RDFS)

Jezik za opisivanje RDF vokabulara. Definiše zajedničke koncepte i relacije. Obezbeđuje jednostavne mehanizme za definisanje ontologija.

Web Ontology Language (OWL) Jezik za predstavljanje znanja. Jedan od ključnih tehnologija Semantic Web-a.

## **28. Triplestore - pojam, osnovne karakteristike i namena.**

TripleStore pripada Graf bazama podataka. Triple Store je dizajniran za smeštanje i pristup RDF tripletima (RDF grafovima). Upitni jezik: SPARQL. Triple Store se često naziva i RDF Database

Semantički web koristi drugačije modele podataka u odnosu na relacione baze.

Triplestore je high-performance storage engine kompatibilan sa RDF (RDFS) i OWL standardima (ontologijama). Većina triplestore rešenja sadrži high-performance rule engine (mašinu za zaključivanje). Mašina za zaključivanje primenjuje skup pravila za zaključivanje na graf (podaci) koji je smešten u bazi, kako bi generisao nove činjenice.

Globalni identifikatori resursa. Jednostavnija integracija podataka. Zaključivanje i izvođenje implicitnih činjenica. Graf model podataka. Pogodan za nepotpune, polustrukturirane i nestrukturirane podatke. Fleksibilna šema. Jednostavno dodavanje novih predikata (odgovaraju kolonama) i novih relacija između klasa. Usklađenost sa Semantic Web standardima.

Način skladištenja podataka: °In memory °Native °Hybrid

Strategije za zaključivanje: °Lančanje unapred °Lančanje unazad °Hibridna strategija