Manje poznati izrazi

Object-relational impedance mismatch – greske koje nastaju kada se pokusava preneti objektno orijentisan model na relacioni model.
CRUD (**crud operations**) – obuhvaraju operacije **C**reate, **R**ead, **U**pdate, **D**elete
RESTFul
Binary safe – niz bitova se posmatra samo kao niz bitova bez ikakvog znacenje kao tekst.
failover – ukoliko dodje do greske u sistemu koja je fatalna postoji protokol koji vrsi transfer kontrole i sistem nastavlja sa radom.
in-memory – obicno se odnosi na radnu memoriju (ram). Nesto se nalazu u ram-u.

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

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

 Struktuirani podaci su zapravo
◦ 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.)

Nedostaci RDBMS
◦ ACID transakcije nisu skalabilne (Horizontalno particionisanje, Neefikasni spojevi)
◦ Transakcije zahtevaju nepotrebnu obradu, odnosno unose dodatni “overhead”
◦ Šema relacionih baza podataka nije fleksibilna

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

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

Distribuirane baze podataka - Više čvorova, jedna baza podataka.

CAP teorema – moguće je zadovoljiti 2/3 zahteva

Podaci koji cine jednu bazu podataka su razdeljeni na više racunara koji su međusobno povezani racunarskom mrežom. Takva baza podataka treba da obezbedi **konzistentnost** (sistem se nalazi u konzistentnom stanju posle svake operacije), **dostupnost** (sistem je uvek dostupan) i **partition tolerance** (sistem funkcioniše cak i u slucaju da ne postoji konekcija između distribuiranih podskupova).

Skalabilnost

Kako kolicina podataka u bazi podataka raste, tako i sistem treba biti u stanju da vecu kolicinu podataka obradi jednako brzo kao i kada obrađuje manju kolicinu podataka. Takva sposobnost sistema se naziva **skalabilnost** i imamo dve vrste skalabilnosti: **vertikalnu** (*scaling up*) i **horizontalnu** (*scaling out*). Kod **vertikalne** se vrši dodavanje resursa jednom cvoru u sistemu (dodavanje CPU ili memorije) ili se vrši migracija sistema na jacu platformu. **Prednost** ovog nacina je njegova brzina i jednostavnost, dok su **nedostaci** cena i kada se dostigne najbolja moguca konfiguracija sistema. Kod **horizontalne** se vrši dodavanje novih cvorova u sistem. **Prednost** ovog nacina je fleksibilnost, dok je **mana** kompleksnost

Scaling Up - Dodavanje resursa jednom 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.
◦ Funkcionalana (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 nalazu 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)

CAP teorema kaže da je u potpunosti moguce zadovoljiti 2 od 3 zahteva (pravi se kompromis oko treceg). Dakle, odustaje se od principa „sve ili ništa" i u tom slucaju treba prepoznati koja su **CAP** (***C****onsistency*, ***A****vailability*, ***P****artition tolerance*) pravila neophodna za funkcionisanje
sistema. Postoje 3 kombinacije pravila:

• **CA** – pravi se kompromis oko *partition tolerance*. Kada dođe do narušavanja topologije mreže, odnosno do particionisanja mreže, sistem se blokira. **Primer**: *dvofazni komit.*

• **CP** – kompromis za *availability*. Pristup pojedinim podacima može biti privremeno onemogucen
ili ogranicen, dok se ostatak sistema nalazu u konzistentom (tacnom) stanju.
Primer: horizontalno particionisane baze podataka na vecem broju servera (*shared databases*).

• **AP** – kompromis za *consistency*. Sistem je dostupan i prilikom narušavanja mrežne topologije. Neki od podataka koje sistem vraca mogu biti privremeno neažurni i takav sistem zahteva strategiju za rešavanje konflikta. **Primer**: DNS, keš, . . .

?? 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) – nije uvek dostupan (*availbality)*
◦ Ukoliko je moguće predvideti opterećenje, moguće je obezbediti A i C rezervisanjem resursa unapred
◦ Nepotrebni resursi (over-provisioning) – A i E

Uporediti ACID i BASE

ACID – Svojstva transakcija se ogledaju u akronimu **ACID** (***A****tomicity*, ***C****onsistency*, ***I****solation*, ***D****urability*). **Atomicnost** govori o tome da se sve naredbe transakcije izvršavaju kao jedinstvena celina ili se uopšte ne izvršavaju. **Konzistentnost** garantuje da se baza podataka nakon transakcije prevodi iz jednog konzistentnog stanja u drugo konzistentno stanje. **Izolacija** govori o tome da efekti transakcije nisu vidljivi u drugim transakcija sve dok se transakcija ne komituje, a **trajnost** govori o tome da su sve izmene nad podacima nacinjene tokom transakcije trajne.

BASE predstavlja CAP varijantu ACID svojstva. Akronim je od ***B**asically **A**vailable*, ***S**oft state*, ***E**ventually consistent*. **Basically Available** garantuje dostupnost u skladu sa CAP teoremom. **Soft state** ukazuje na to da se stanje sistema može promeniti i bez ikakve operacije nad njim, zbog svojstva *Eventually consistent*. **Eventually consistent** svojstvo nam govori da ce sistem vremenom postati konzistentan, pod uslovom da u tom periodu ne bude upita nad bazom.
Dakle, ACID forsira konzistentnost podataka dok BASE prihvata da ce se konflikti desiti.

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), Not Only SQL

Tipična primena:
◦ **Velike količine podataka** - Za skladištenje podataka se koristi distribuirana arhitektur a
◦ **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. Taksonomija 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 klonovi. Rasuta, distribirana 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-strukturiani dokument čija je struktura razumljiva DB. (U key/value podaci nisu razumljivi bazi) 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 oriented baze podataka predstavljaju softver koji se koristi za pribavljanje, skladistenje i upravljanje *document-oriented* ili *polu-struktuiranim podacima*.

Document store su jedna od NoSQL baza podataka. Document store su zapravo key/value baze u kojima je *value* polustruktuirani dokument cije je struktura razumljiva bazi.

Sve informacije se nalaze u samom dokumentu, nema povezanih tabela. Ne moraju svi dokumenti da imaju istu strukturu.

Dokument store znaci da: (6)

- **Objekti (podaci) se cuvaju kao dokumenti**. Tako da ne postoji object-relational impedance mismach.
- **Dokumenti mogu da budu jako kompleksni**. Citav objektni model moze da se ucita/snimi odjednom, nema potrebe da za velikim brojem CRUD atomicnih operacija.
- **Dokumenti su nezavisni**. Povecavaju se preformanse i smanjuju problemi za konkurentan pristup podacima.
- **Otvorenog su formata.** Najcesce se koriste neki siroko koriscen 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.

Baza je indeksirana po vrednosti kljuca kako bi pristup dokumentima bio optimalan, osim toga postoji i API ili upitni jezik za pretrazivanje za konkretnu bazu podataka.
Vecina ovih baza obezbedjuje **RESTFul api**, kao i **http servere** koji podrzavaju standardne http zahteve (put, get, post, delete).

Pretrazivanje: Osim pristupa dokumentima pomocu kljuca, one sadrze API pomocu kog je moguce detaljnije pretrazivati bazu po nekim atributima objekata. Ne postoji standardni jezik za pretragu.

Za organizaciju dokumenata koristi se: KTH&M

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

Jedna od osnovnih karakteristika dokument store-a je **eventual consistency**, to omogucava korisnicima da menjaju dokumenta, izmene nisu vidljive u istom trenutku svim klijentima ali postaju

vidljive u nekom trenutku. Tako da se povremeno javlja nekonzistentnost podataka. Cilj je povecanje skalabilnosti i dostupnosti podataka.

Koriste se za: (primena)

- **Dinamički podaci** - CMS (Content Management Systems) i *CRM* (Customer Relationship Management) objekti koje korisnici mogu da menjaju i prilagođavaju sopstvenim potrebama.
- **Polustruktuirani 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 podrske za ACID svojstva)
- Kada je potrebno koristiti SQL (potreba za velikim brojem spojeva)

Neki predstavnici dokument baza podataka su: **MongoDB**, CouchDB, RaptorDB, RavenDB,..

MongoDB

Koristi JSON za predstavljanje podataka. Interno se dokumenti skladiste u BSON formatu.

Osobine:

- Puna podrska za indeksiranje dokumenata
- Visok nivo skalabilnosti
- Visoka dostupnost
- Podrska za replikacije
- Kompleksni upiti
- Map/reduce podrska.

Osnovni tipovi podataka: number, string, array, boolean, object, null.

Indeksi

Mongo indeks predstavlja strukturu podataka koja omogucava brzo lociranje dokumenata na osnovu vrednosti odredjenih polja u dokumentu.
Mongo indeksi se kreiraju

- na nivou kolekcije dokumenata
  i
- nad poljem dokumenta
- delom polja
- nad vise polja (kompozitni indeksi)

MongoDB indeks je implementiran kao struktura B-stabla. Oni sluze za brze pribavljanje dokumenata, ali usporavaju operacije azuriranja. Svaka mongo operacija, pretraga ili azuriranje, koristi samo jedan indeks. Query optimizer bira indeks koriscenjem empirijskih podataka.

Za svaku kolekciju u MongoDB-u postoji **_id** jedinstveni indeks identifikator kolekcije. Za njega se najcesce koristi **ObjectID** vrednost.

SUUS

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

Ugnjezdeni indeksi – Moguce je koristiti *prefiksnu kombinaciju* za predragu po njima, smer sortiranja ima kljucnu ulogu ako ih je potrebno sortirati nakon pribavljanja

Unique index – sprecava dupliranje vrednosti u poljima koja su indeksirana. Podrazumevano, indeski nisu jedinsteni.

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

MongoDB obezbedjuje mehanizme za **agregaciju podataka**. Oni prihvataju kolekciju dokumenata koja se prosledjuje kroz pipeline, tj. niz medjusobno povezanih operatora($project, $match, $limit, skip, unwind, group, sort..).

Dokument je **ogranicen** na velicinu od 16MB. Za pamcenje vecih dokumenata koristi se *GrigFS* mehanizam. Za *GrigFS* se koriste **2 vrste kolekcija**:

- files - sadrzi metapodatke o dokumentu
- chunks - sadrzi delove dokumenta.

Replikacija – omogucava sinhronizaciju izmedju veceg broja mongodb instanci. Skup instanci izmedju kojih je uspostavljen taj mehanizam se zove *MongoDB replication set*. Jedna instanca je primarna, druge su sekundarne. Upisivanje se vrsi u primarnu instancu, a nakon toga se sekundarne asinhrono updejtuju. *Failover mehanizam* - Ukoliko se primarna kopija izgubi, ostale masine glasaju na kojoj masini ce da se nadje nova primarna instanca.

Sharding mehanizam – on sluzi za particionisanje kolekcije i njihovu distribuciju na vise instanci (shards). Ovo ima cilj da poveca kapacitet sistema(scale out).
Osobine: Particionisanje se vrsi na osnovu kljuca dokumenta, particija obuhvata dokumente koji se nalaze u nekom opsegu vrednosti kljuca, kada particija preraste kapacitat instance ona se deli i rasporedjuje na vise manjih particija. Automatski se balansira kolicina podataka na razlicitim instancama. Ovaj proces distribuiranja podataka je potpuno transparentan za korisnike. Opterecenje se rasporedjuje na veci broj instanci cime se povecava ukupni kapacitet sistema.

Map/Reduce – *Programski model* za obradu velike kolicine podataka.
Namena – distribucija obrade na veci broj cvorova, omogucava paralelnu obradu velike kolicine podataka na distribuiranim cvorovima, obradjuju se podaci koji se nalaze u file sistemu ili u bazi podataka. Odvija se u 2 koraka:

1. Map – master cvor prihvata ulaz, deli ga na manje potprobleme i distribuira ih worker cvorovima, kad oni obrade prosledjuju master cvoru.
2. Reduce – master cvor prikuplja i spaja podatke od vorker cvorova i pravi rezultat originalnog problema.

Postoje implementacije u razlicitima jezicima, popularna je *Apache Hadoop*.

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

Column-store baze podataka

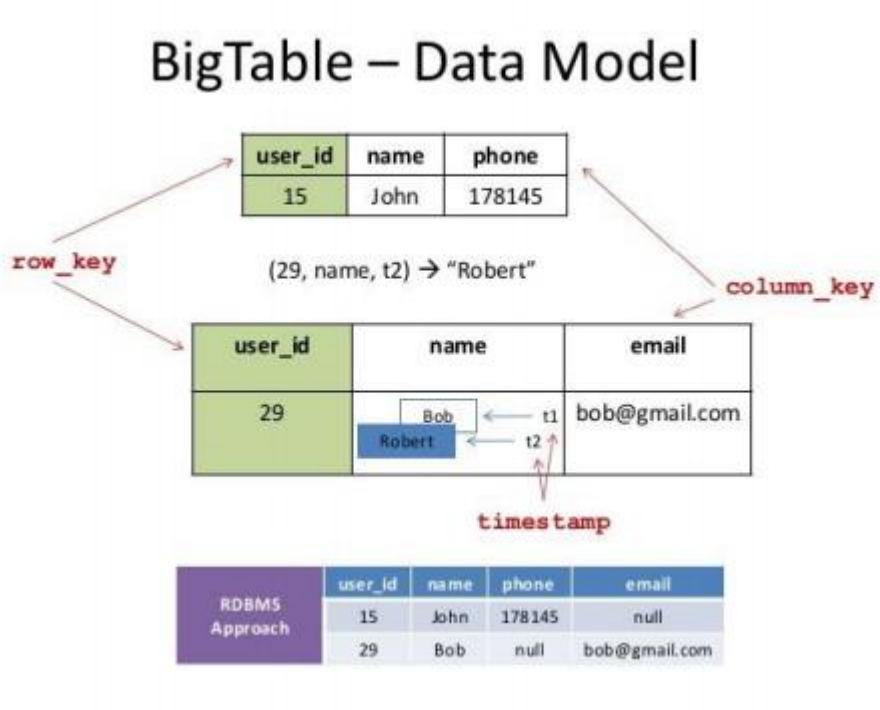
Predstavljaju klonove BigTable-a.
To je u stvari rasuta, distribuirana multi dimenzionalna sortirana mapa.

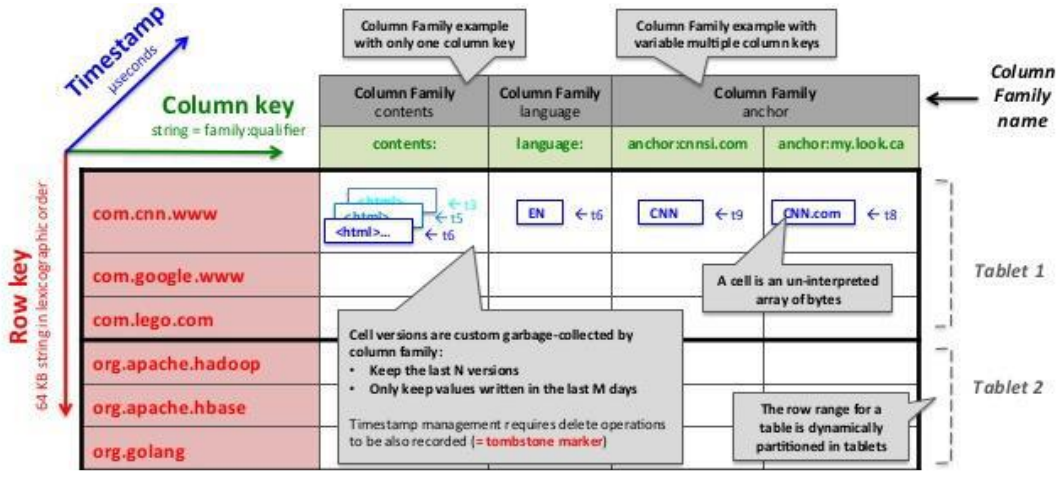
Konceptualno:

- Jedna beskonacno velika tabela.
- Svaka vrsta moze imati razlicite kolone (po broju i sadrzaju)
- Retko posednuta tabela |vrste| * |kolone| > |vrednosti|

BigTable

BigTable - Rasuta, distribirana, perzistentna multi-dimenzionalna sortirana mapa. Dimenzije (row, column, timestamp). Vrednost je nestruktuirani niz bajtova. Karakteristike: hibridno row/column skladiste, Jedna master kopija, Verzije.





Model podataka

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

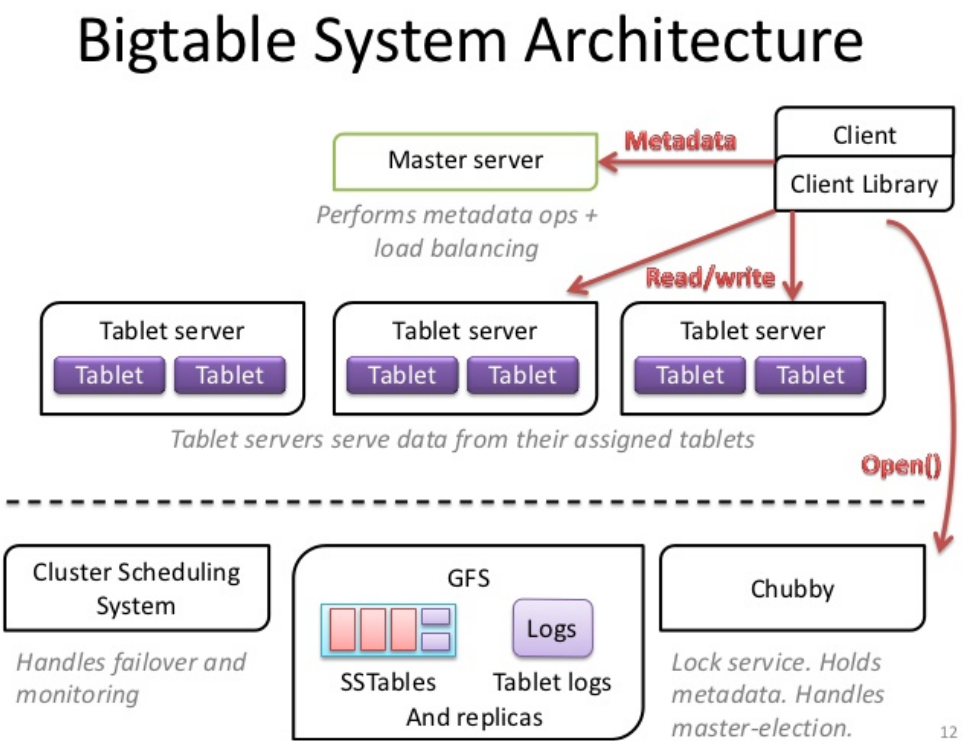
Kolone, Familije kolona ^: Ključevi kolona su proizvoljni stringovi. Neograničeni broj kolona. Familiji kolona se pridružuje informacija o tipu podataka. Podaci u familiji kolona se čuvaju i kompresuju zajedno. Kontrola pristupa je implementirana na nivou familija kolona.

Timestamp: Svaka ćelija može da ima više vrednosti. Timestamp se moze rucno promeniti/dodeliti.

Tableti: Opseg vrsta je dinamički particionisan 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)

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



Arhitektura:

- Podaci se cuvaju u Google file system-u (GFS)
- Chubby (distriburani 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

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

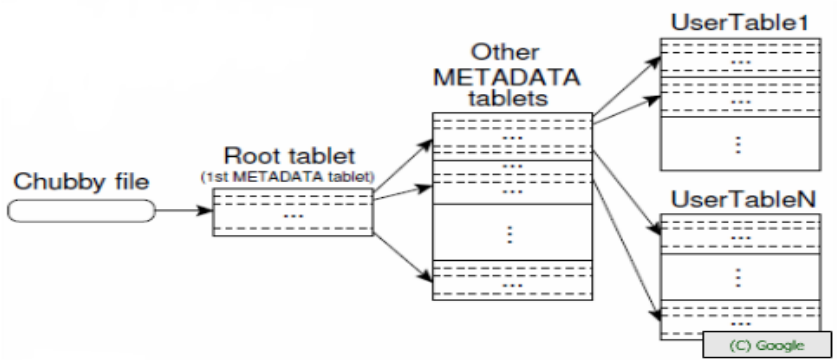
Lokacija tableta:

- B+ stablo

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

Struktura B+stabla. Klijenti keširaju lokacije podataka.
Rooot metadata tablet se cesto nalazi na zasebnom serveru kako ne bi predstavljao usko grlo.

Na slici je BigTable workflow :



Pristup tabletu

SSTabel (sorted strings table)

Perzistentna, uredjena, nepromenljiva mapa kljucevi-vrednost

Ključevi i vrednosti su stringovi
Sadrži sekvencu blokova i indeks bloka
Kompresija se vrši na nivou bloka
Dvofazna kompresija, 10:1

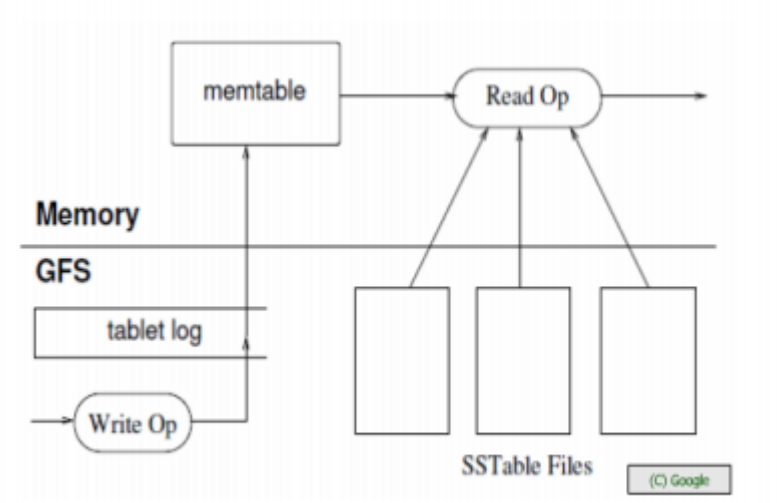
Ažuriranje:

- Ažurira se redo-log
- Ažurira se 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:

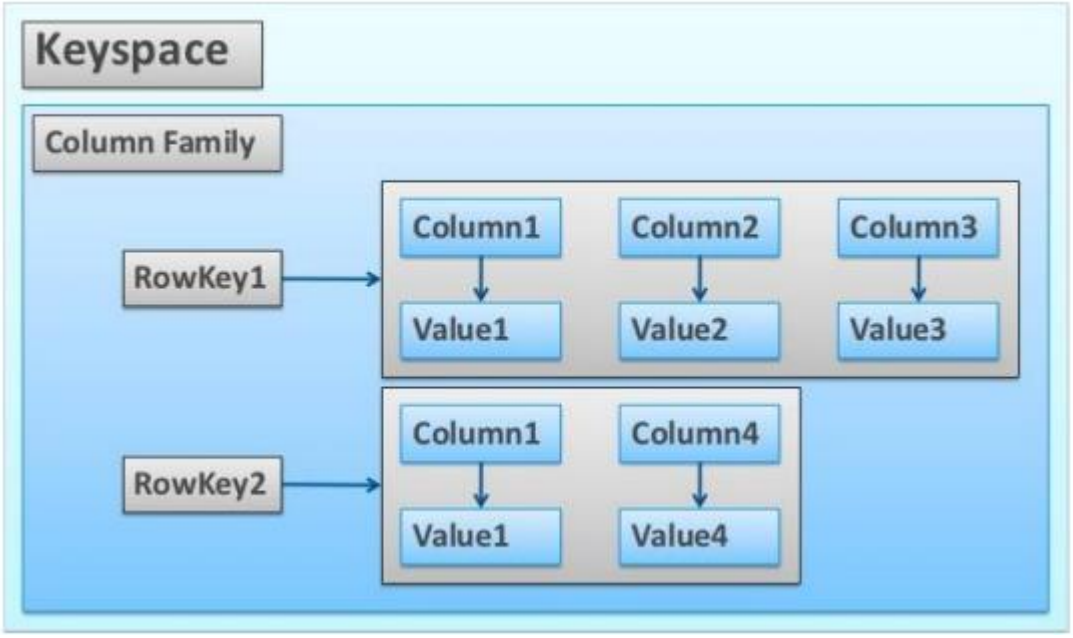
- Izvršava se nad memtable i svim SSTable
- Pristup memtable je izuzetno efikasan (podaci su unapred sortirani)
- Koriste se **Bloom filteri** –mape koje definišu da li određena SSTable sadrži odgovarajuću ćeliju (vrsta, kolona) -drastično smanjen broj pristupa fajl sistemu



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. Jedno od najpopularnijih NoSQL resenja (FB, insta..) Prednosti: skalabilnost, dostupnost, performanse.

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



Relational Model	Cassandra Model
Database	Keyspace
Table	Column Family (CF)
Primary key	Row key
Column name	Column name/key
Column value	Column value

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 podatak sam po sebi. (Generalno arhitektura BigTable)

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

- Faktor replikacije (koliko kopija ima svaki podatak)
- Faktor distribucije (kako se distribuira)
- Koje column family sadrzi

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.

- Osnovna struktura podataka:

Tipovi vrsta: (np. U odnosu na kolone)

- 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:

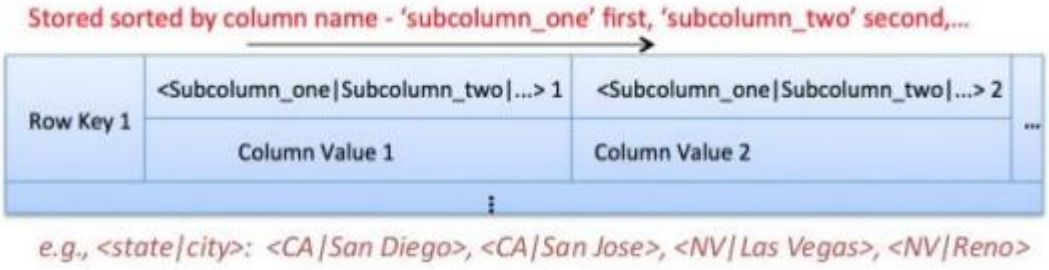
[Keyspace][Column family][Key] **[SuperColumn][Subcolumn]**

U pojedinim situacijama je boljekoristiti kompozitne ključeve.

1

- Superkolona predstavlja mapu podkolona.

- Korišćenje kompozitnih kolona umesto superkolona (često se preporučuje).



Model podataka

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 moguda menjaju poziciju u prstenu zbog balansiranja opterećenja.

Operacije

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 prvo dodaju u commit log -> zatim u memtable -> i na kraju u SSTable. (slicno azuriranju)

Read (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, Koristese Bloom filteri.

Kompakcije(== BigTable)

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

Podaci na internetu su polustruktuirani ili nestruktuirani, kompleksne i neregularne strukture, pa relaciona baza nije pogodna za njihovo predstavljanje.

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*, tj. Svaki element, cvor, moze da ima vezu s nekim drugim, a da to nije opste pravilo za sve cvorove tog tipa.

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.** (Neo4j koristi indekse za lookup pretragu grafa, u cilju pronalazenja cvorova i relacija na koje ce se primeniti grafovske operacije. Indeksi se koriste da obezbede jedinstvenost vrednosti pojedinih svojstava. Podrazumevani indeks je Lucene-open source biblioteka koja obezbedjuje mehanizme za indeksiranje i pretrazivanje podataka. Indeksi mogu biti Exact indeksi – indeksira se kompletna vrednost svojstva, indeksiraju se numericke ili string vrednosti , sva svojstva mogu da se indeksiraju kao string vrednosti. ILI Full text indeksi – indeksiraju se sve reci u tekstualnom podatku)

Podaci iz grafbaza 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 pogodne za razvoj ad-hoc aplikacija ili aplikacija koje brzo evaluiraju.

Pogodne za operacije tipa: obilazak grafa, traženje najkraćeg puta između dva čvora, provera da li postoji put između dva čvora, detektovanje paterna, određivanje suseda i sl.
Poređenje sa drugim bazama

Relacione baze podataka su optimizovane *za agregaciju podataka*.
Graf baze su optimizovane za *veze izmedju podataka*.

Key-value skladista podataka su optimizovana za *jednostavne lookup pretrage*.
Graf baze za obilaske *povezanih podataka*.

Key-value su optimizovana za organizaciju podataka u *strukture stabla*.
Graf baze omogucavaju organizaciju podataka u *razlicite kompleksne strukture* povezanih podataka.

Neo4J

Graf baza podataka + Lucene Index.
Property graf

Full ACID (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. Relacija se uvek može obići, bez obzira na smer. Svaka relacija ima tip, tj labelu.

Svojstva predstavljaju key-value parove. Vrednosti svojstava mogu bit 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.

RestApi, za reprezentaciju podataka se podrazumevano koristi JSON.

Cypher –deklarativni upitni jezik za grafove. Omogućava CRUD operacije nad Neo4J grafovima. Izgrađen je na elementima SQL-a (START, MATCH, WHERE, RETURN, CREATE, DELETE, SET, FOREACH, WITH)

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 svoju kolekciju atributa i njihovih vrednosti. Podatak može biti predstavljen kao BLOB objekat pričemu 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 odključa do ključa.
◦ Memorijska efikasnost – predstavljaju se samo atributi čijevrednosti postoje (kod relacionih baza podataka svaki podatakamora da ima sve atribute).
◦ Ne postoji problem object-relational impedance mismatch.
◦ Korišćenje relacione baze podataka u sprezi sa sistemom zakeširanje 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

Poznata resenja: Redis, MemcachedDB, DynamoDB...

Upotreba key/value

Document-oriented baze predstavljaju specijalan slucaj key/value baza
Neke graf baze interno koriste key/value paradigmu za skladistenje uz dodatak relacija

Redis(Remote Dictionary Server)

Redis predstavlja najpopularniju implementaciju key/value paradigme. Value moze biti prestavljen koriscenjem razlicitih struktura podataka.

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

- **Snapshoting** – kompletan skup podataka iz memorije se periodično snima na disk
- **Apend-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.

Zahvaljujuci in memory implementaciji Redis nudi odlicne performanse u odnosu na RDBMS sisteme kod kojih svaka transakcija mora da se snimi na disk da bi bila potvrdjena. Medjutim takva impelementacija dovodi do toga da izmene nad podacima koji nisu perzistentni ili nisu sinhronizovane mogu biti izgubljene.

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.

Redis koristi binary safe kljuceve, prazan string je validan kljuc. Treba izbegavati isuviše duge ključeve koji mogu da dovedu do pada 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

Ide objasnjenje svih tipova i operacija detaljno...

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 strane nekog drugog klijenta. Transakcija je atomicna, procesiraju se sve komande ili nijedna. Ne postoji rollback. Ako dodje do greske u jednoj komandi, ostale iz iste transakcije ce se izvrsiti.

Redis implementira jednostavnu Publish/Subscribe paradigmu.

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 implementiratio stablo replikacije sa originalnim master serverom u korenu stabla (single-rooted replication tree). Redis slave server može da prihvata i wite 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 podatakaizmeđu većeg broja instanci Redis servera.

//////////////////////

Dynamo

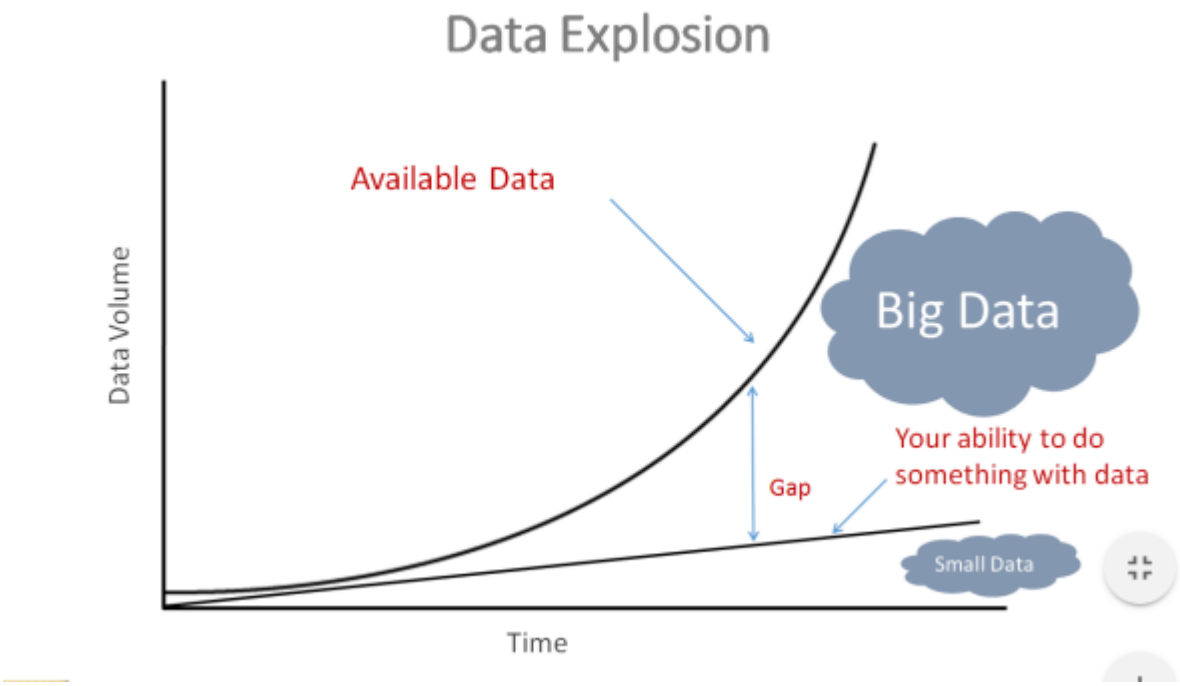
Dynamo is the name given to a set of techniques that when taken together can form a highly available key-value structured storage system or a distributed data store. It has properties of both databases and distributed hash tables (DHTs). Amazon DynamoDB is "built on the principles of Dynamo". It was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

Struktuirani podaci (Tradicionalni podaci u relacionim bazama, ERP i CRM, ACID transakcije), Polustruktuirani i nestr.podaci(Format podataka nije konzistent, Veoma je tesko definisati record/slog u tradicionalnom smislu, Dominantan tip podataka na web-u, Zhtevaju novi pristup i nove tehnike za obradu)



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 (2.5×1018) 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.

Hadoop NIJE:
Relaciona baza, OLTP sistem, zamena za postojece DataWarehouse sisteme, struktuirano skladiste podataka, fajl sistem, zamena za programsku logiku.

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 claster-u nije konstantan

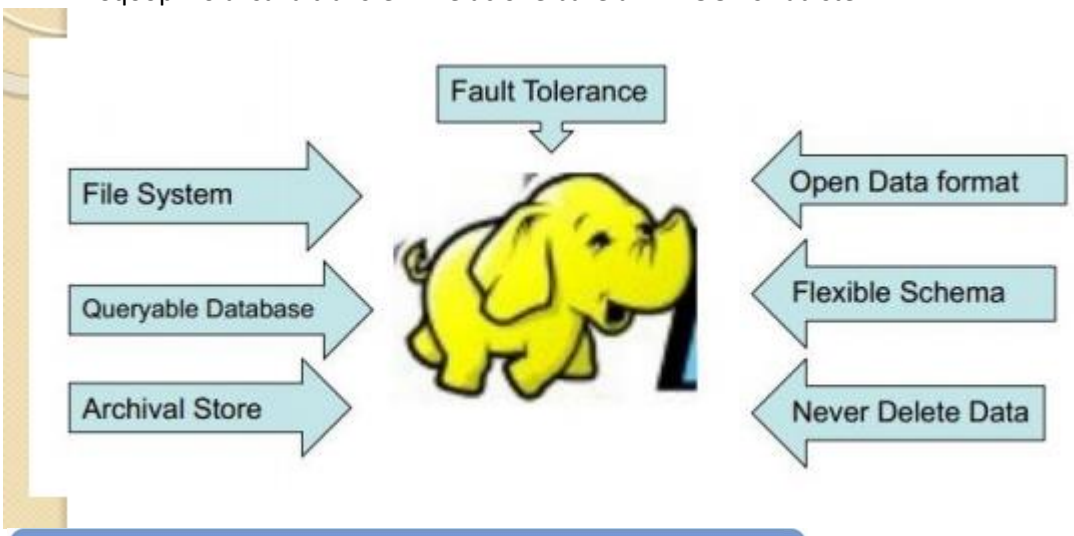
Razlika izmedju **hadupa** i **relacione baze** podataka:

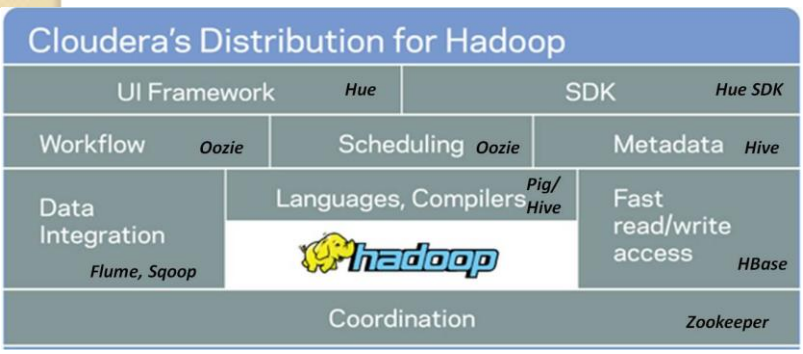
- Brz upis podataka vs Brzo citanje podataka
- Scale out vs scale up
- Key/value vs table
- Definise se kako obradjivati podatke vs definisu se potrebni podaci
- Fleksibilnost/ agilnost vs standardi/ upravljanje

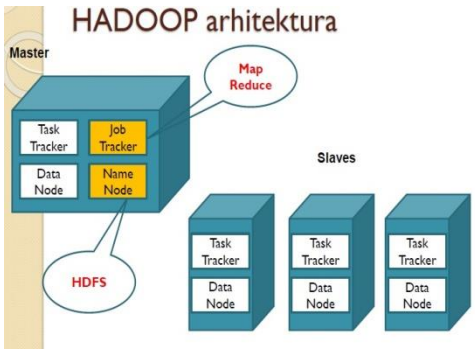
15. HADOOP ekosistem – osnovni elementi, arhitektura i gde se primenjuje

HADOOP ekosistem se sastoji od kolekcije softverskih komponenti:

- HADOOP Common
- Hadoop Distributed File System (HDFS)
- MapReduce
- HBASE
- HIVE – relaciona baza razvijena koriscenjem Hadoop ekosistema
- Pig – jezik namenjen pojednostavljenju Map-Reduce job-ova
- Sqoop – olaksava transfer iz relacione baze u HADOOP skladiste





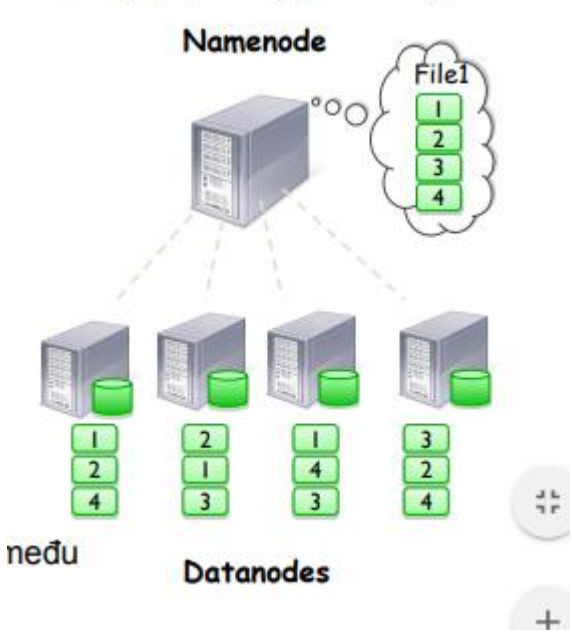


Primena: Search, Log processing, Clickstream, Recommendation Systems, Sentiment analysis, Data Warehouse, Video and Image Analysis

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

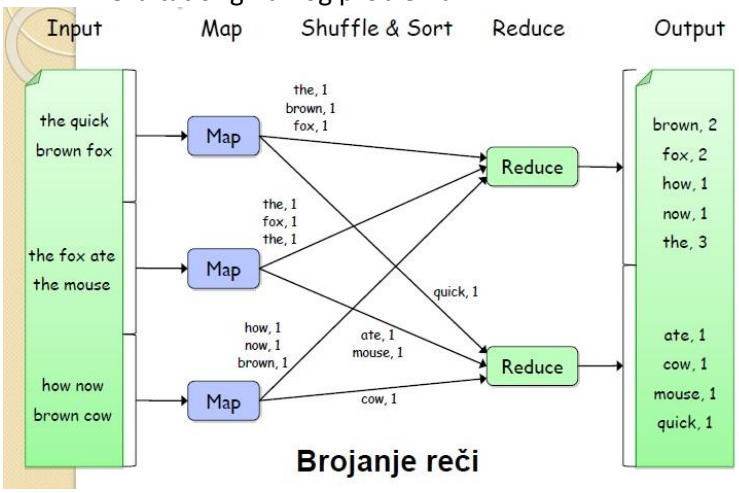
17. Pojam MapReduce tehnike

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
- **Shuffle –**sort and merge, tranzicija izmedju Map i Reduce faze
- **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 manjiproblem, 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)

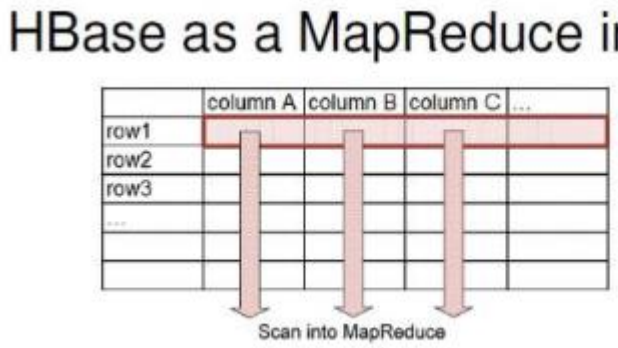
3. **Sporo izvršavanje task-a**
- Kopija taska se pokreće na drugom čvoru
- Prihvata se rezultat taska koji se prvi završi

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

HBASE - Bazira se na BigTable specifikaciji.
Svojstva:

- Podaci se cuvaju sortirani
- automatsko particionisanje i rebalansiranje i reparticionisanje
- tolerancija na greske i otkaze (*HDFS odrzava tri replike podataka*).

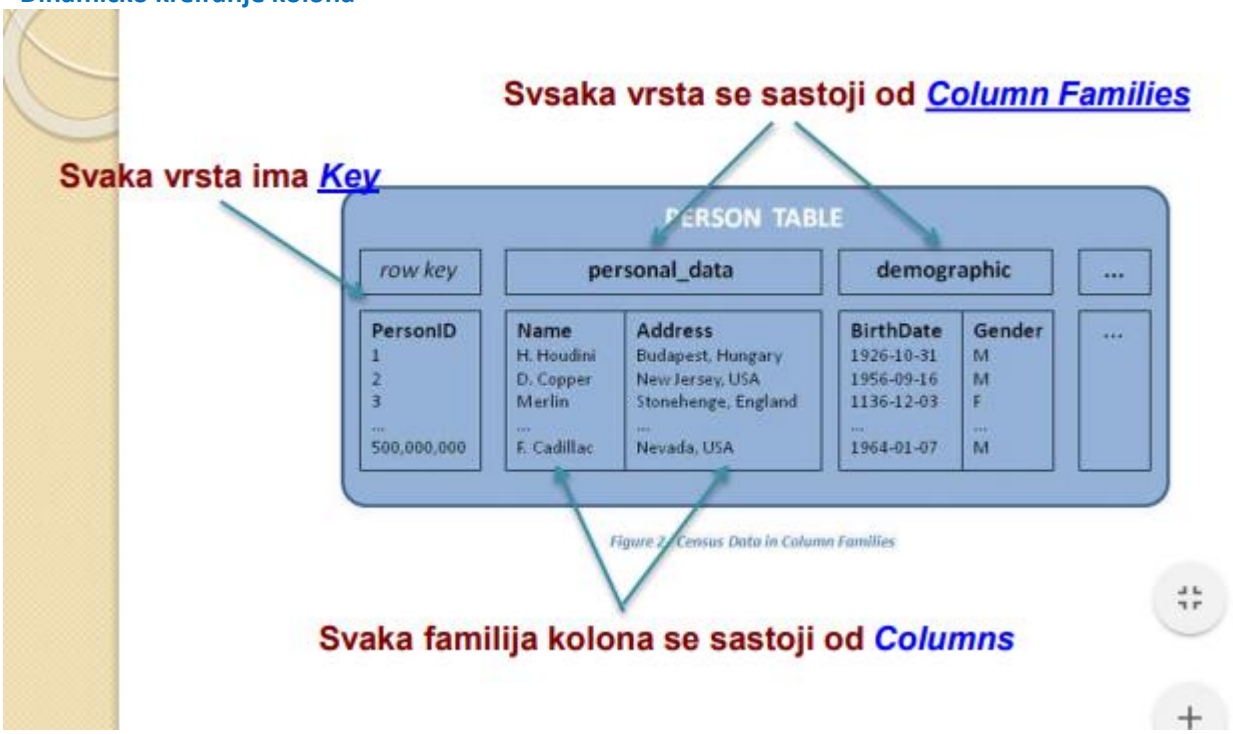
Operacije: Lokacija Hstore/tablet, Write, Read, Kompakcija.
Hbase-MapReduce: Hbase moze da bude ulaz u MapReduce job, svaka vrsta predstavlja ulazni slog job-a, MapReduce omogucava da se paralelizuje obavljanje operacija nad vecim brojem vrsta.



//////////////////////

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.

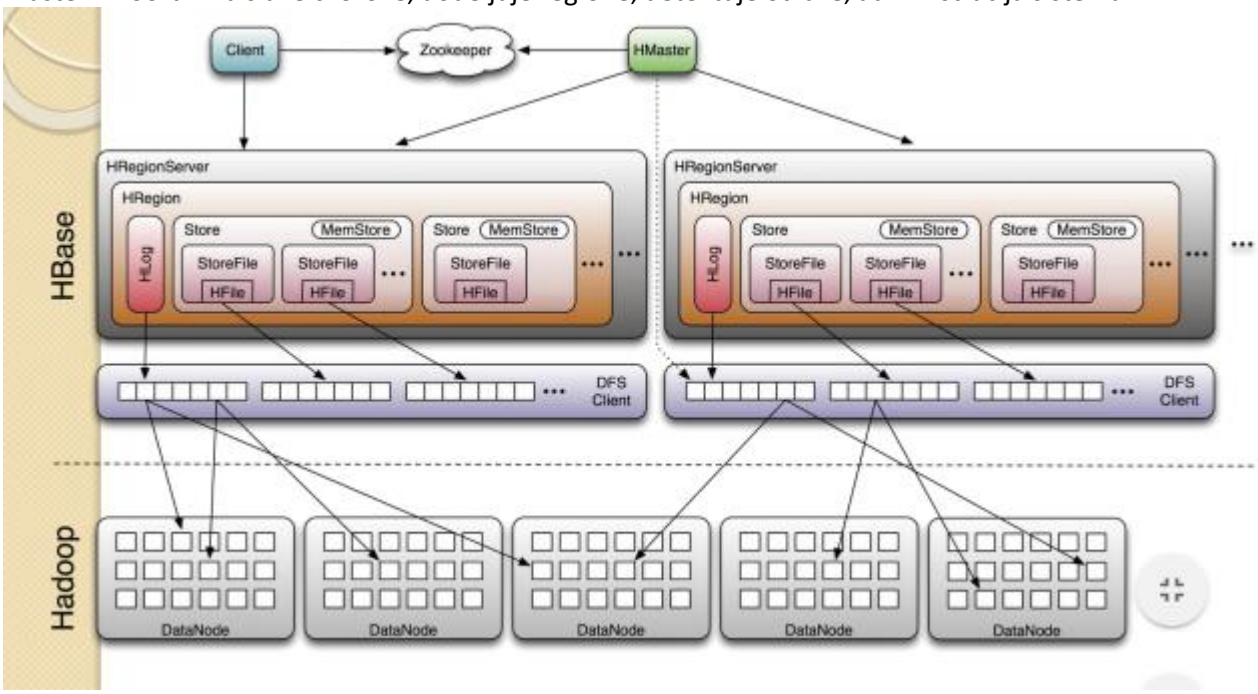
Tabele: velike, retko popunjene podacima, slabo struktuirane
◦ **Sastoji se** od vrsta, jedinstveni ključevi vrsta
◦ **Vrste** imaju različiti broj kolona
◦ **Kolone** su grupisane u familije
◦ **Dinamičko kreiranje kolona**



Model podatka: Row key – Column family – Value - Timestamp
Hbase sema se sastoji od nekoliko Tables, svaka tabela od nekoliko Column Families(one se inace cuvaju u posebnoj datoteci HFile)
Hbase ima Dynamic Columns, imena kolona se nalaze sa podacima uokviru celija, razlicite celije mogu da imaju razlicite kolone.
Svaka kolona familija se particionise po horizontali – regioni. Regioni odgovaraju HDFS blokovima.
Postoji jedan master Hbase master, vise region servera, Hregion Server i Hbase klijenti

Region – podskup vrsta table, automatsko particionisanje

RegionServer, vise slave servera – upravlja regionima podataka, obezbedjuje podatke za read/write operacije (koristi log)
Master – koordinira slave cvorove, dodeljuje regione, detektuje otkaze, administracija sistema



Kada koristiti HBASE: VEEEEEELIKE STVARI

1. **Nije pogodan za 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: R'N'P

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

Vremenska serija predstavlja niz vrednosti neke veličine, pribavljanih u sukcesivnim vremenskim intervalima, gde su intervali često jednaki.

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

1. **Opšti trend f** - 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 s** - 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 c**- 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 epsilon** - Ovakve varijacije se nekada mogu nazivati iregularnim fluktuacijama. Te promene, iako su slučajne u prirodi, mogu nagovestiti kontinualnu promenu u trendovima u predstojećem periodu.

Modeli vremenskih serija : AMM

- Aditivni,
- Multiplikativni,
- Mesoviti (Xt = f*c*(s+epsilon-1))

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.
- **Efikasan pristup podacima u proizvoljnom intervalu**

Sistem koji smešta sortirane podatke u sekvencijalnom redosledu -> (ko-lokacija) na istim mestima, što ide prirodno uz vremenske serije podataka, će učiniti te podatke dostupnim za brzo i efikasno čitanje.
 Forsiraju se CR- a ne CRUD operacije

Osnovne karakteristike:

1. **Performanse i skalabilnost -** Dobro projektovana TSDB omogućava lako skaliranje sistema tako da on može da podrži veliki broj informacija koje moze da skladisti
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. To znači da se uglavnom visoko precizni podaci kratko čuvaju u bazi (npr. vrednost neke promenljive u intervalima od jedne sekunde), tim podacima se konstantno smanjuje rezolucija (downsample) i raznim matematičkim funkcija agregacije (aggregate) se objedinjuju u tačke sa većim periodima
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.

Neki primeri baza: RRDTool, OpenTSDB, KairosDB, RiakTS, InfluxDB, ..

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

Primeri

- **RDDTool** - RRDtool je napisan u programskom jeziku C. Podaci se smeštaju u bazi koja je implementirana kao kružni bafer.
- **OpenTSDB** - Napisan u Javi, 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. Implementiran koriscenjem jezika Go.

Arhitektura: TKCI

- **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. Kontinualni upiti omogućavaju smanjenje rezolucije (downsampling) vremenskih serija podataka.
Politika zadrzavanja - Svi podaci u bazi koji su stariji od zadatog vremena (npr. 23 časova i 40 minuta) će biti konstantno brisani.

22. Objasnite pojam pretraživanja informacija

Pretraživanje informacija predstavlja aktivnost pribavljanja relevantnih resursa iz kolekcije resursa na osnovu zadatog kriterijuma. Resurs najčešće predstavlja nestrukturiani document. Pretraga moze biti bazirana na metapodacima ili na full text ili drugim indeksima. Najpoznatiji sistem za pretragu danas jesu internet pretrazivaci.
Kod ove prve pretrazuju se samo polja koja opisuju sam dokument(naslov, autori, datum izdavanja, zanrovi i sl).
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 kolko su relevantni za dati upit (query).
- **Analiza** – mogućnost pretvaranja bloka teksta u posebne normalnizovane tokene, na osnovu kojih se kreira indeks za pretragu.

Invertovani indeks – *indeksna struktura* kod koje se sadrzaj dokumenta, reci ili brojevi, mapira na lokaciju u okviru dokumenta ili skupa dokumenata. Ovo omogucava izuzetno brzu pretragu.
Strutkura inverovatnog indeksa:
- **Vokabular** – lista reci koje se pojavljuju u dokumentu ili skupu dokumenata
- **Pojavljivanja** – lista koja sadrzi informacije o svakoj reci iz vokabulara

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. (u offline rezimu)
- **Procesiranje upita** - interpretacija korisnikovog zahteva odnosno pitanja na osnovu čega se izvrsava upit (query). (u real time rezimu)

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.

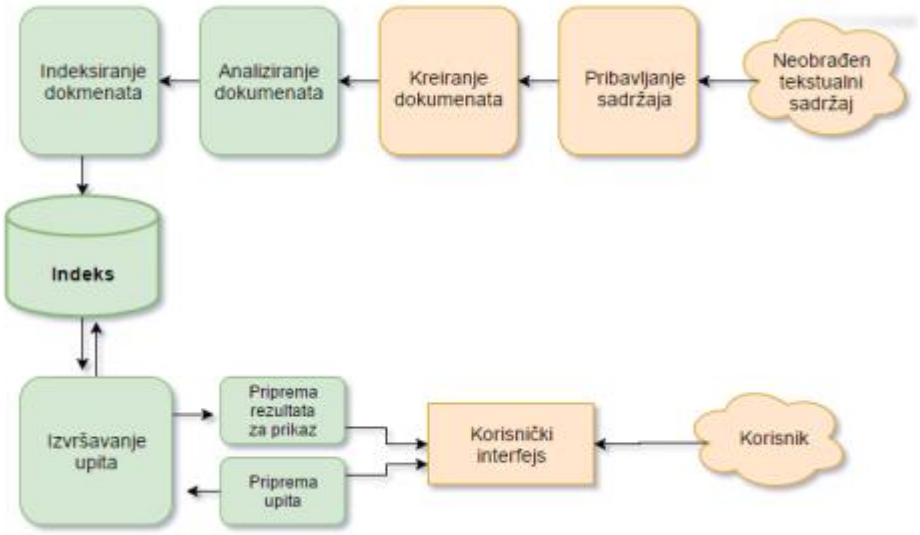
Struktura invertvanog indeksa:

- **Vokabular** (vocabulary) – lista svih reči koje se pojavljuju u dokumentu ili skupu dokumenata.
- **Pojavljivanja** (Occurencies) - lista koja sadrži informacije o svakoj reči iz vokabulara (dokumenit u kojima se reč pojavljuje, poyicija 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 pretaživanje 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: BVP

- **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.

  Lucene biblioteka kombinuje *vektorski model i Bulov model.*

- **Probabilistički model** – U ovom modelu se računa verovatnoća da dokument zadovoljava upit *(koristi se u biblioteci Xapian).*

- Prednosti korišćenja:
  - Brzina
  - Jednostavan API
  - Konkurentno indeksiranje i pretraživanje
  - Inkrementalno indeksiranje
  - Near real-time performanse

 Nedostaci:
- ◦ Ne podržava spojeve
- ◦ Nema delimičnog ažuriranja dokumenata
- ◦ Ažuriranje dokumenta se svodi na operaciju brisanja I dodavanja novog dokumenta.

25. Elasticsearch – namena i osnovne karaktersitike

Resenje zasnovano na Lucene biblioteci. NoSQL **document store** platofrma 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 balansiranej opterećenja prilikom izvršavanja upita.

// ako treba informativno :
Klaster (Cluster) – Kolekcija od jednog ili više servera (čvorova) koji zajedno čuvaju korisničke podatake i omogućuju indeksiranje i pretragu podataka u čvorovima koji se nalaze u okviru klastera. Čvor (Node) – Čvor je server na kome se čuvaju podaci i koji je deo klastera.
Indeks (Index) - Indeks je kolekcija dokumenata koji imaju neke zajedničke karakteristike. Na pojam indeksa se može gledati kao na pojam baze podataka u relacionom modelu.
 Tip (Type) – U okviru indeksa mogu postojati više različitih tipova. Tip predstavlja logičko particionisanje indeksa. Generalno, dokumenta koja pripadaju istom tipu, sadrže ista polja tj. imaju istu strukturu (mapiranje). U koliko posmatramo paralelu sa relacionim modelom, tip bi odgovarao pojmu tabele.

Dokument (Document) – Dokument je osnovna informativna jedinica koja se može indeksirati. Svaki dokument se nalazi u JSON formatu.
 Indeksiranje dokumenata (to Index a document) – Pod indeksiranjem dokumenta, podrazumeva se smeštanje dokumenta u indeks (index) odnosno pribavljanje, obrada, analiza dokumenta I ažuriranje indeksa.
 Shard (particije) – Indeks može da sadrži velike količine podataka, koji mogu da prevazilaze hardverske mogućnosti čvora na kome se nalaze, Elasticsearch deli indeks na mnogo manjih delova – particije. Na particiju može da se gleda kao na samostalni, u potpunosti funkcionalni indeks koji može da se nalazi na bilo kom čvoru u klasteru.

Shard (particije) ◦ Mehanizam podele indeksa na particije obezbeđuje horizontalno skaliranje, distribucija i paralelizacija pretrage na nivou particija. ◦ Za sam proces podele indeksa na particije i agregacija dokumenata iz particijaprilikom upita, je zadužen Elasticsearch i taj mehanizam je transparentan korisniku.

 Replike - Da ne bi došlo do otkazivanja sistema i gubljenja podataka usled otkaza u mreži, odnosno da bi se omogućila dostupnost sistema, svaki shard može da ima jednu ili više replika (replica shards). Replika shard se nikad ne nalazi na istom čvoru kao i njen original (primary shard).

Za svako tekstualno polje dokumenta moguće je navesti analizator.

26. Objasniti pojam semantičkog Web-a

Semantički web je unapređenje postojeđeg 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 koristit mašine (računari, aplikacije) ne samo za prikaz već i za deljenje, automatizaciju i integraciju između različitih aplikacija.Metapodaci obezbeđuju mehanizme za definisanje značenja podataka. Semanticki web je jos uvek vizija ka kojoj se tezi.

Semanticki web je kljucna komponenta web 3.0 ili GGG – Giant Global Graph.

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.

Ima jos nekih detalja...

27. Objasniti pojam ontologija

Ontologija predstavlja eksplicitnu specifikaciju konceptualizacije.
Danas imaju veliku primenu u oblasti računarstva za opisivanje znanja. Računarski čitljive definicije osnovnih koncepata 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.

Ontologije obicno imaju dve razlicite komponente:
-Imena za bitne koncepte i veze u domenu (biljojed je koncept ciji su clanovi one zivotinje koje jedu samo biljke)
- Background znanje o domenu (nijedna instanca ne moze da bude i biljojed i mesojed)

Resource Description Framework (RDF)
Namena:
- Definisanje podataka i metapodataka
- Definisanje struktura skupova podataka
- Definisanje relacija između delova podataka

RDF iskaz se satoji iz tri dela, subjekat, predikat-svojstvo, objekat-vrednost.

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

RDF + RDF Schema = RDF(S)

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

Prica o OWL, Sparql pa onda Triplestore

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, polustruktuirane i nestruktuirane 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

Non native
- Hybrid
  - On top of SQL engine
  - On top of NoSQL engine

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

I jos detalja…