Univerzitet u Beogradu
Matematički fakultet

Nerelacione baze podataka
NoSql

Student	Nemanja Veljković	1007/2012
Predmet	Metodologija naučnog i stručnog rada	
Školska godina	2012/2013	
Nastavnik	dr Vladimir Filipović	
Datum	10.06.2013	

Sadržaj

1 Uvod 2

2 Skalabilnost 2

3 NoSql baze podataka 3

4 Vrste NoSql baza podataka 5

4.1 Uređeno sortirana skladišta orijentisana ka kolonama 5

4.2 Ključ/vrednost baze podataka 6

4.3 Baze podataka orijentisane ka dokumentima 6

4.4 Grafovske baze podataka 8

5 Zaključak 9

6 Literatura 10

1. Uvod

Kao i većina novih i dolazećih tehnologija, NoSql je obavijen oblakom straha, neizvesnosti i sumnje. Svet programera se podelio u 3 kategorije kada je u pitanju NoSql:
1 – Oni koji ga obožavaju – koriste ga, poboljšavaju i prate novine u svetu NoSql-a
2 – Oni koji ga kritikuju – uglavnom se fokusiraju na njegove nedostatke i trude se da pokažu loše strane NoSql-a
3 – Oni koji ga ignorišu – mogu se podeliti u dve grupe: prvi čekaju da ova tehnologija sazri, dok drugi veruju da je ova tehnologija samo trenutni hir kojeg će uskoro neka nova tehnologija izbaciti sa tržišta

U savremenom razvoju softvera termin nerelacione baze podataka se odnosi na sisteme za upravljanje kolekcijama podataka koje:

- nemaju strogu statičku strukturu podataka
- nemaju iscrpnu proveru uslova integriteta
- ne koriste upitni jezik Sql

Naziv **NoSql** nastao je 1998. godine upravo zato što je Sql simbol relacionih baza podataka. Bilo je pokušaja da se zovu i NonRel baze, ali je NoSql preovladao. Danas se tumači kao "*Not only Sql*". Bez obzira na bukvalno značenje, NoSql se koristi danas kao opšti termin za sve baze podataka i skladišta podataka koje ne prate popularne i dobro-utvrđene principe relacionih baza podataka i često se odnose na velike skupove podataka kojima je potrebno brzo i efikasno pristupiti i menjati ih na Vebu. To znači da NoSql nije jedinstveni proizvod, čak ni jedinstvena tehnologija. Predstavlja klasu proizvoda i koncepata u vezi skladištenja podataka i njihovog upravljanja.

2. Skalabilnost

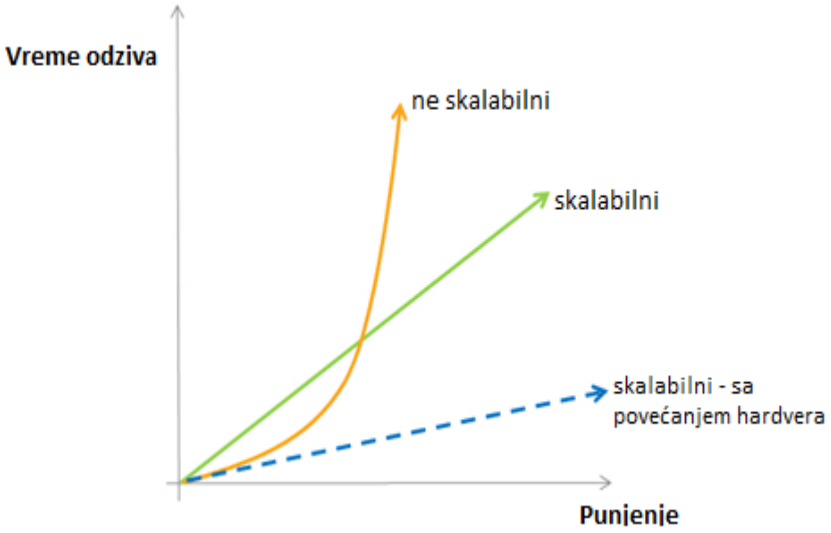
Skalabilnost je mogućnost aplikacije da podnese povećanje zahteva i broja korisnika, a da sama aplikacija ne mora da se menja. Što je aplikacija skalabilnija ona će lakše podneti povećan protok podataka. Cilj kojim teže svi projektanti sistema jeste da se postigne linearnost u brzini odgovora na zahtev i količine podataka sa kojima se manipuliše.

Kada govorimo o samom hardveru postoji horizontalna skalabilnost i vertikalna skalabilnost. **Vertikalna skalabilnost** je kada je aplikacija smeštena na jednom serveru, a na povećan protok se reaguje tako što se serveru dodaje memorija, jači procesor, nova jezgra ili dodatni hard disk.

Horizontalna skalabilnost je idealnije rešenje, posebno za velike sisteme. Dodavanjem novih čvorova sistem nastavlja da radi kao do sada samo sa novim igračem (čvorom) u timu. Čvor predstavlja jedan server.

Kada Veb aplikacija dođe do stadijuma da povećan broj podataka sa kojima se manipuliše utiče na brzinu odgovora na zahtev, tj na učitavanje stranica, može se reagovati na više načina:

1. Uložiti gomilu novca u kupovinu hardvera koji će moći da se nosi sa novonastalom situacijom.
2. Misliti na vreme i dizajnirati samu aplikaciju tako da bude skalabilna, a to se postiže tako što se na probleme odgovara rešenjima koja podižu performanse.



MapReduce model[1] patentiran od strane Google-a predstavlja jednu od najpopularnijih metoda za procesiranje takvih podataka. To je model paralelnog programiranja koji omogućava distribuirano procesiranje velikih količina podataka smeštenih na klasterima računara. Ova ideja potiče iz sveta funkcionalnog programiranja, gde ako imamo listu [1, 2, 3, 4] i na nju primenimo neku mapirajuću funkciju koja kreira novu listu [2, 4, 6, 8], dok prethodna lista ostaje neizmenjena. Takodje postoji i koncept redukujuće funkcije, kod koje ako na primer primenimo funkciju zbira na listu [2, 4, 6, 8] dobijamo kao rezultat zbir svih članova liste, tj. 20.

Ova ideja se koristi i pri obradi velikih količina podataka, s tim što je izmenjena da radi sa kolekcijama torki ili ključ/vrednost parovima. Mapirajuća funkcija se primenjuje na svaki ključ/vrednost par u kolekciji i generiše novu kolekciju, na koju se kasnije može primeniti redukujuća funkcija.

3. NoSql baze podataka

Nedostatak relacionih baza podataka predstavlja relativno visoka cena čitanja zbog neredundantnosti i stroge strukture podataka. Otežano je distribuiranje pre svega zbog konzistentnosti podataka. Skupa je promena strukture zbog povezanosti strukture sa upotrebom i optimizacijama.

[1] http://en.wikipedia.org/wiki/MapReduce

Današnje veb aplikacije susreću se sa mnogim problemima. Konkurentni korisnici u ogromnom broju koji se teško određuje, dnevna proizvodnja terabajta i petabajta podataka. Obrada podataka je otežana, jer korisnici u svakom trenutku vrše izmenu tih podataka. Opterećenje je neravnomerno sa nepredvidivim porastom, neravnomerno velika dinamičnost, neprekidno dodavanje novih mogućnosti, promena postojećih komponenti.

Nema jedinstvene definicije, ali se može reći da je nerelaciona baza podataka je baza podataka koja ne počiva na relacionom modelu podataka ili ga se bar ne drži čvrsto. Lako se distribuira, horizontalno je skalabilna tj. lako podnosi značajne promene šeme.

Iako koncepti mogu da podsećaju na RBP (katalog-tabela, vrednost-red,...) zapravo postoje značajne razlike. Struktura vrednosti obično nije strogo predefinisana, obično je netipizirana, ne insistira se na neredundantnosti, praktično retko ima prirodnih ključeva, ne postoji referencijalni integritet.

Druge česte karakteristike su da nema statičku shemu, lako se replicira, poseduje jednostavan API, omogućava ogromne količine podataka, počiva na skupu principa BASE [2](eng. *Basically Available, Soft state, Eventual consistency*), a ne ACID (eng. *Atomicity, Consistency, Isolation, Durability*).

Ovde je bitno navesti i CAP teoremu koja kaže da je nemoguće u distribuiranim računarskim sistemima da se istovremeno omoguće sve tri garancije:
1) Konzistentnost – da svi čvorovi vide iste podatke u isto vreme
2) Raspoloživost– svaki zahtev prima odgovor da li je uspešno izvršen ili ne
3) Prihvatanje razdvojenosti - sistem nastavlja rad bez obzira na poruku o neuspehu na nekom delu sistema

Suštinski raspoloživ: Ovo ograničenje tvrdi da sistem garantuje raspoloživost podataka veći deo vremena. Na svaki zahtev će se vratiti odgovor. Međutim, ta potvrda zahteva može dovesti do neuspeha prilikom dobijanja zahtevanih podataka ili može ostati u nekonzistentnom stanju tokom obrade.

Nekonzistentno stanje: Baza podataka ne mora biti konzistentna u svakom trenutku. Stanje sistema se menja tokom vremena, pa čak i kada nema unosa, već do promene dolazi usled konvergentne konzistencije. Tako da je stanje sistema uvek “soft”.

Konvergentna konzistencija: Dodatkom novog čvora u sistem sadržaj se može replicirati na njega. Ne postoji garancija da će u svakom trenutku svi čvorovi sadržati identične kopije podataka. Teži se vremenskoj tački u kojoj će svi čvorovi sadržati konzistentne podatke.

[2] prevod ovih termina preuzet od profesora Ivana Lukovića sa Fakulteta Tehničkih Nauka u Novom Sadu

4. Vrste NoSql baza podataka

4.1 Uređeno sortirana skladišta orijentisana ka kolonama

Guglov Bigtable [3] predstavlja model u kome se podaci čuvaju uređeni prema kolonama, što predstavlja kontrast u odnosu na RSUBP (Sistemi za upravljanje relacionim bazama podataka), gde se podaci čuvaju uređeni u redovima. Ovako se podaci smeštaju mnogo efikasnije zato što ako nema podataka za datu kolonu oni prosto neće biti upisani, dok kod RSUBP se upisuju NULL vrednosti.
Svaka jedinica može da se posmatra kao skup parova ključ/vrednost, gde se svaka jedinica identifikuje sa primarnim identifikatorom, koji se često naziva redni ključ. Jedinice podataka su raspoređene u uređeno-sortiranom poretku prema rednom ključu.

Primer:
Imamo prostu tabelu koja sadrži polja ime, prezime, zanimanje, post_kod, pol, i podatke:

ime: Jovan
prezime: Jovanović
post_kod: 11000
pol: muški,

ime: Petar
post_kod: 18000

Redni ključ prvog podatka je 1, a drugog 2. Podaci se čuvaju sortirano po kolonama, tako da će podatak sa rednim ključem 1 biti upisan pre podatka sa rednim ključem 2.

Podaci u Bigtable i njegovim klonovima su smešteni u sekvencijalnom redosledu. Kada podaci napune jedan čvor, oni se dele u više čvorova. Podaci su sortirani ne samo na jednom čvoru već i na svim čvorovima koji čine jedan veliki sekvencijalni skup. Podaci i dalje postoje u redosledu tolerantnom na greške gde se održavaju 3 kopije istih podataka. Većina klonova održava distribuirani sistem datoteka za čuvanje podataka na disku. Distribuirani sistem datoteka omogućava da se podaci čuvaju na klasterima.

Hbase [4] je popularno, open-source[5] , uređeno prema kolonama, skladište podataka koje je modelovano na osnovu ideja Google-ovog Bigtable-a. Podacima u Hbase-u se može upravljati pomoću Hadoop MapReduce alata[6]. Implementiran je u Javi, kao upitni jezik može se koristiti Hive [7](nalik na Sql) . Izdat je pod Apache License version 2. licencom, a od poznatih kompanija ga koriste Facebook, Yahoo!, Adobe, itd.

[3] http://en.wikipedia.org/wiki/BigTable
[4] http://hbase.apache.org
[5] http://en.wikipedia.org/wiki/Open-source_software
[6] http://hadoop.apache.org
[7] http://hive.apache.org/

4.2 Ključ/vrednost baze podataka

Heš mapa ili asocijativni niz je najjednostavnija struktura podataka koja može čuvati skup ključ/vrednost podataka. Takve strukture podataka su popularne zato što je potrebno O(1) vremena za pristup podacima. Ključ iz para je jedinstvena vrednost u skupu pomoću koje se brzo pristupa podacima.
Ključ/vrednost parovi su različitih tipova: neki čuvaju podatke u memoriji, a neki pružaju mogućnost perzistencije podataka na disku. Ključ/vrednost parovi se mogu distribuirati u klasteru čvorova.
Jedna od poznatijih baza ovog tipa je Oracle-ova **BarkeleyDB** [8]gde su i ključ i vrednost nizovi bajtova. Engine baze podataka ne pridaje značenje ključevima ili vrednostima, već uzima parove iz nizova bajtova i vraća iste pozivu. BarkeleyDB dopušta da se podaci keširaju u memoriji i prebace na disk kada narastu. Takođe, postoji i indeksiranje ključeva za brži pregled i pristup.
Drugi tip ključ/vrednost parova koji je u opštoj upotrebi je keš. Keš pruža presek stanja u memoriji najkorišćenijih podataka u aplikaciji. Cilj keša je da redukuje ulazno-izlazne operacije diska. Keš sistemi mogu biti rudimentarne strukture mapa ili robustni sistemi sa politikama isteka podataka iz keša.

Ovi tipovi ključ/vrednost parova pružaju strogu konzistenciju podataka. Međutim postoje i one koje ističu dostupnost ispred konzistencije u distribuiranom raspoređivanju. Najpoznatiji predstavnik te klase je Amazonov Dynamo koji obećava izuzetnu dostupnost i skalabilnost i predstavlja okosnicu Amazonovog sistema za skladištenje podataka. Najvažniji koncept ovog sistema je eventualna konzistentnost. Eventualna konzistentnost podrazumeva da mogu postojati mali intervali nekonzistentnosti između repliciranih čvorova kada se podaci ažuriraju između peer-to-peer čvorova.

Najpoznatija open-source implementacija ovog sistema je **Appache Cassandra**[9] razvijena od strane Facebook-a, a kasnije donirana Appache fondaciji. Implementirana je u Javi, a upitni jezik je trenutno u fazi razvoja. Izdata je pod Apache License version 2. licencom, a od poznatih kompanija je koriste Facebook, Digg, Reddit, Twitter, itd.

4.3 Baze podataka orijentisane ka dokumentima

Skladišta orijentisana ka dokumentima nisu sistemi za upravljanje dokumentima. Reč dokument sugeriše labavo struktuirane skupove ključ/vrednost parova u dokumentima, obično JSON (JavaScript Object Notation). JSON jeste tekstualni standard namenjen razmeni podataka u formi čitljivoj i razumljivoj i za čoveka i za računar. JSON format je izveden iz JavaScript jezika, ali sam format nije zavistan od bilo kog konkretnog programskog jezika. MongoDB dokumente skladišti u formatu koji se naziva BSON, i koristi ga za razmenu

[8] http://en.wikipedia.org/wiki/Berkeley_DB
[9] http://cassandra.apache.org

podataka sa aplikacijama na mreži. BSON format se zasniva na JSON-u i nastao je od skraćenice "Binary JSON". Baza tretira dokument kao celinu i izbegava podelu dokumenta na sastavne ključ/vrednost parove. Na nivou kolekcije, ovo omogućava skladištenje različitih skupova dokumenata zajedno u jednu kolekciju.

Najpoznatija open-source implementacija ovog tipa je **MongoDB**[10]. Kreirana je od strane 10Gen kompanije, implementirana u C++. Upitni jezik je napravljen u JavaScript-u, ali podseća pomalo na Sql. Izdata je pod GNU Affero GPL licencom, a od poznatih kompanija je koriste FourSquare, SAP, MTV, SourceForge, Github, itd.

Funkcionalnosti MongoDB:

Indeksiranje - indeksi obezbeđuju visoke performanse izvršenja operacija kod najčešće korišćenih upita. MongoDB indeks predstavlja strukturu podataka koja omogućava brzo lociranje dokumenata na osnovu vrednosti određenih polja u dokumentima.

Agregacija podataka – MongoDB pruža funkciju agregate() koja se koristi za agregaciju podataka i koja se poziva nad određenom kolekcijom. Koriste se i sledeći operatori: sort (sortiranje dokumenata), skip (ignoriše određeni broj dokumenata na početku kolekcije), match (filtriranje dokumenata na osnovu vrednosti polja), group (grupisanje dokumenata), limit (definiše broj dokumenata koji prolaze kroz filter), project (određuje polja iz dokumenta koja će biti uključena u rezultat) i unwind (transformiše niz).

Mehanizam replikacije – ovaj mehanizam omogućuje sinhronizaciju podataka između većeg broja MongoDB instanci. Postoji samo jedna primarna instanca, a sve ostale su sekundarne. Replikacija ima zadatak da obezbedi redundantnost, poveća dostupnost i olakša neke administrativne zadatke poput pravljenja rezervnih kopija. MongoDB obezbeđuje i mehanizam ako otkaže primarna instanca, mehanizmom glasanja bira se nova primarna instanca.

Mehanizam deljenja - ovaj mehanizam omogućuje particionisanje kolekcije dokumenata i njihovu distribuciju na veći broj instanci. Ideja koja stoji iza mehanizma deljenja je da se poveća kapacitet sistema.

Map/Reduce – omogućuje paralelnu obradu većih količina podataka.

Ad-hoc upiti - Dinamički (ad hoc) upiti omogućuju rad sa MongoDB-om na sličan način kao sa RDBMS-om. Samim tim, ova karakteristika je pogodna pri prelasku sa nekog RDBMS sistema na MongoDB.

GridFS mehanizam - ovaj mehanizam omogućuje pamćenje velikih dokumenata. Konkretno, MongoDB može da radi sa dokumentom veličine 16MB. Zbog ovog ograničenja koristi se GridFS mehanizam koji document deli u niz komada. Za čuvanje velikih dokumenata koriste

[10] http://www.mongodb.org

se dve kolekcije: kolekcija “files” koja sadrži metapodatke o dokumentu i kolekcija “chunks” koja sadrži delove dokumenta. Veličina delova na koje se dokument deli obično je 256KB.

4.4 Grafovske baze podataka

Grafovske baze podataka koriste grafovske strukture sa čvorovima (koji predstavljaju entitete), granama (koje povezuju čvorove) i svojstvima (predstavljaju atribute) za reprezentaciju i čuvanje podataka. Po definiciji grafovska baza je svaki sistem kod kojeg svaki elemenat sadrži direktni pokazivač na susedni element i nije potreban nikakav pregled indeksa. Ove baze su moćan alat za upite poput na primer izračunavanje najkraćeg puta između dva čvora u grafu.

Jedna od poznatijih baza ovog tipa je FlockDB[11]. Kreirana je od strane Twitera i koristi se za skladištenje liste susednih pratilaca na Twiteru. Implementirana je u Scala programskom jeziku i izdata pod Apache License version 2. licencom.

U nastavku sledi tabelarni prikaz poređenja različitih NoSql rešenja i Relacionih sistema za upravljanje bazama podataka:

	performanse	skalabilnost	fleksibilnost	kompleksnost	big data
ključ/vrednost	*visoke*	*visoka*	*visoka*	*nema*	*da*
or. ka kolonama	*visoke*	*visoka*	*srednja*	*mala*	*da*
dokument	*visoke*	*promenljiva*	*visoka*	*mala*	*ne*
graf	*promenljive*	*promenljiva*	*visoka*	*visoka*	*ne*
RSUBP	*promenljive*	*promenljiva*	*visoka*	*srednja*	*ne*

[11] https://github.com/twitter/flockdb

5. Zaključak

NoSql ne predstavlja rešenje za sve probleme i ima svojih nedostataka. Suština je da se NoSql rešenja ponašaju dobro prilikom skaliranja, kada podaci rastu u velikim količinama i potrebno je da budu distribuirani na više čvorova u klasteru. Procesiranje velikih količina podataka predstavlja jednako veliki izazov i zahteva nove metode.

Trenutna generacija programera je odrasla sa relacionim bazama podataka i navikavanje na NoSql je teško, jer je usađen već jedan način mišljenja, koji se sada u potpunosti menja. To znači da kao programer, čovek treba da proučava NoSql i razume ga dobro, pre nekog kritikovanja, čega u današnje vreme ima dosta. Takođe, mnogo ideja u NoSql-u se mogu veoma dobro iskoristiti prilikom rešavanja problema velike skalabilnosti u drugim tipovima aplikacija.

Nerelacione baze podataka neće u bliskoj budućnosti izbaciti relacione. To uopšte nije i njihov cilj. One dopunjuju RSUBP i u zavisnosti od potreba će se odlučiti koji sistemi će se koristiti, dok je u velikim kompanijama, poput Facebooka-a, praksa da se koriste i jedne i druge.

6. Literatura

(1) Shashank Tiwari, *Professional NoSql*, Wrox Press (2011)

(2) Saša Malkov, Univerzitet u Beogradu - Matematički fakultet, *Materijal sa predavanja DOBP* http://poincare.matf.bg.ac.rs/~smalkov

(3) Ivan Luković, Univerzitet u Novom Sadu, Fakultet Tehničkih nauka, *Materijal sa predavanja SBP* http://www.acs.uns.ac.rs/sr/sbp

(4) http://nosql-database.org/

(5) http://docs.mongodb.org/manual/

(6) http://www.10gen.com/nosql

(7) Slobodna encoklopedija, *Wikipedia* http://www.wikipedia.org