Seminarski rad

- Sistemi za upravljanje bazama podataka -

Tema: *Baza podataka Redis*

Student:
Marko Oblak

Mentor:
Aleksandar Stanimirović

Sadržaj

UVOD

Redis sistem baza podataka je besplatan projekat otvorenog koda (open source) koji je nastao 2009. godine. Kompletno je razvijen korišćenjem ANSI C jezika (portabilnost na različite platforme). Redis je akronim od Remote Dictionary Server (srp. udaljeni rečnički server).
Server čuva podatke u primarnoj memoriji, što omogućava veoma brze operacije čitanja i pisanja u bazu podataka. Zbog gubitka podataka pri gašenju servera, postoje mehanizmi kojima se podaci cuvaju u sekundarnoj memoriji (trajna memorija) na neki od dva načina – čuvanjem svih podataka iz baze kad se uslovi ispune (broj upisa u bazu, zakazano vreme itd.) ili čuvanjem svake od izvršenih naredbi nad bazom.

Redis je NoSQL sistem baza podataka koji organizuje podatke u ključ – vrednost parove (Key/Value Stores). Ključ je niska (uređeni niz simbola) koji jedinstveno određuje jedan zapis, dok vrednost moze biti tipa: String, Lista, Skup, Uređeni skup, Heš i HyperLogLog.

Redis sistem baza podataka trenutno koristi veliki broj kompanija kao što su Twitter, GitHub, StackOverflow, Pinterest, Instagram, Snapchat. Može se koristiti kao baza podataka, kao sistem za keširanje podskupa podataka iz primarne baze podataka, keširanje sesije, kao implementacija redova zadataka, kao posrednik poruka itd.

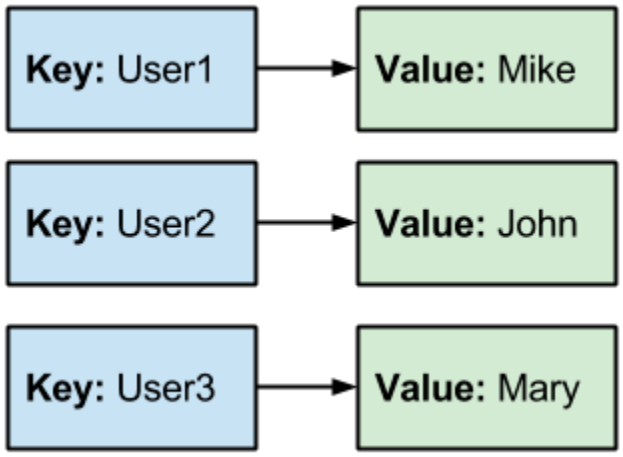
-NoSQL-

Termin se upotrebljava za sve nerelacione baze podataka (non-RDBMS). Primenjuju se za skladištenje velike količine podataka (distribuiranja arhitektura) – Google, Amazon, Facebook (10k – 100k servera), kada je potreba za upitima velika (kod RDBMS mala efikasnost). Pozitivne karakteristike su što su fleksibilne, skalabilne, jeftine (osnova, infrastruktura je skupa), prilagođene potrebama Web aplikacija, a loše strane su da tehnologija jos uvek nije stabilna, ne postoje zajednički standardi, loša podrška za transakcije, loša podrška za pretraživanje

podataka, zahteva promenu načina razmišljanja i takođe, vrlo je teško naći dva identična scenarija primene.

-Key/Value Stores-

Ključ/vrednost paradigma za pristup podacima obezbeđuje dobre performanse i dostupnost podataka. Nema duplikata – jedna vrednost jedan ključ. Ključevi i vrednosti mogu biti kompleksni objekti.



Slika 1. *Ključ/Vrednost*

Prednosti su efikasne pretrage po ključu (predvidive performanse), jednostavna distribucija podataka na klasteru, fleksibilnost šeme (struktura podataka se moze razlikovati od ključa do ključa), memorijska efikasnost (predstavljaju se samo atributi čije vrednosti postoje, za razliku od relacionih baza gde svaki podatak mora da ima sve atribute)..
Nedostaci su da ne postoje kompleksni upiti, nepostojanje stranih ključeva, za kreiranje spojeva između različitih podataka odgovorna je aplikativna logika, standardizacija..

Redis tipovi podataka

Redis sistem za upravljanje bazama podataka podržava nekoliko tipova podataka koji liče na standardne strukture podataka u programskim jezicima. Prilikom rešavanja različitih problema veliku pažnju treba posvetiti odabiru pravog tipa podatka.

Redis string

Redis string se sastoji iz sekvence karaktera i može sadržati tri vrste vrednosti: tekst (xml, json, html ili sirovi tekst), brojeve (cele brojeve i brojeve u pokretnom zarezu) i binarne vrednosti. Maksimalna veličina je 512 MB. Neke od svrha korišćenja su:

- Keš mehanizmi - za čuvanje teksta ili binarnih podataka u Redisu;
- Brojanje - ako vrednost tipa String može da se interpretira kao ceo broj u dekadnom sistemu ili kao razlomljeni broj, Redis će to detektovati i biće moguća njhova obrada pomoću naredbi poput inkrementacije i dekrementacije.

Takođe je omogućeno čitanje i pisanje niski u druge niske.

Tipične naredbe za rad sa tipom podatka Redis String date su u tabeli 1.

Ime naredbe	Opis naredbe
SET	Postavlja vrednost na dati ključ.
GET	Pribavlja vrednost sa datog ključa.

Tabela 1. *Naredbe za rad sa Redis stringom*

Redis lista

Lista u Redisu je implementirana kao jednostruko povezana lista elemenata. Elementi liste su tipa String, i sortirani su prema redosledu dodavanja u listu. Budući da je ona implementirana kao jednostruko povezana lista, a ne kao indeksirani niz, operacija dodavanja elemenata u veoma dugu listu je veoma brza, što je njena glavna prednost. Operacije dodavanja i čitanja prvog i poslednjeg elementa iz liste izvršavaju se u konstantnom vremenu O(1), dok je složenost operacije pristupanja n-tom elementu O(n). Stoga, ukoliko je operacija pristupanja elementima u sredini frekventnija, postoje pogodniji tipovi podataka od liste. Tipične primene Redis liste su:

- Kod društvenih mreža za čuvanje poslednjih n poruka/komentara/slika;
- Kod međuprocesne komunikacije za smeštanje poruka.

Tipične naredbe za rad sa listama date su u tabeli 2.

Ime naredbe	Opis naredbe
RPUSH	Postavlja vrednost u element na kraju liste.
LPUSH	Postavlja vrednost u element na početku liste.
RPOP	Uklanja i vraća element sa kraja liste.
LPOP	Uklanja i vraća element sa početka liste.
LINDEX	Vraća element sa zadate pozicije.
LRANGE	Vraća elemente iz liste u zadatom intervalu, uključujući i elemente sa početne i krajnje pozicije intervala.
LTRIM	Uklanja elemente iz liste koji se ne nalaze u intervalu zadatom u argumentima.

Tabela 2. *Naredbe za rad sa Redis listom*

Redis skup

Redis skup predstavlja neuređenu kolekciju niski. Elementi u skupu moraju biti jedinstveni, odnosno nisu dozvoljeni duplikati. Skup je implementiran kao heš tabela, što implicira da se dodavanje, uklanjanje i pretraga vrednosti vrši u konstantnom vremenu. S obzirom na to da su elementi skupa neuređeni, elementi se dodaju u skup i uklanjaju iz njega po vrednosti. Tipične primene skupova su:

- Filtriranje ili grupisanje podataka - koristeći operacije preseka, razlike i unije skupova. Na primer pronaći sve letove koji kreću iz jednog, a završavaju u drugom mestu;
- Proveravanje pripadnosti grupi.

Tipične naredbe za rad sa skupovima date su u tabeli 3.

Ime naredbe	Opis naredbe
SADD	Dodaje elemente u skup.
SCARD	Vraća broj elemenata u skupu.
SDIFF	Vraća razliku skupova.
SINTER	Vraća presek skupova.
SISMEMBER	Vraća 0 ukoliko je dati element član skupa ili 1 ukoliko dati element nije član skupa.
SMEMBERS	Vraća sve elemente skupa.
SUNION	Vraća uniju skupova.

Tabela 3. *Naredbe za rad sa Redis skupom*

Redis uređeni skup

Uređeni skup (engl. sorted set, zset) predstavlja uređenu kolekciju jedinstvenih vrednosti tipa String. Svakom elementu skupa je dodeljen realan broj - rang (engl. score). Elementi u uređenom skupu su sortirani na sledeći način:

- ako elementi A i B imaju različiti rang onda: A > B ako je A.rang > B.rang, A < B ako je A. rang < B. rang,
- ako elementi A i B imaju isti rang onda se na njih primenjuje leksikografsko poređenje.

Operacije su sporije nego kod neuređenih skupova zbog prisustva ranga koga je potrebno upoređivati, i logaritamske su složenosti. Ovaj tip podatka je posebno pogodan kada je potrebno pristupati elementima čiji je rang poznat. Tipične primene su:

- Prikaz liste čekanja u realnom vremenu;
- Prikaz najboljih rezultata u igri, prikaz korisnika sa rezultatom sličnim zadatom korisniku;
- Automatsko završavanje reči.

Tipične naredbe za rad sa uređenim skupovima su date u tabeli 4.

Ime naredbe	Opis naredbe
ZADD	Dodaje elemente u uređeni skup.
ZCARD	Vraća kardinalnost uređenog skupa.
ZCOUNT	Vraća broj elemenata uređenog skupa u datom intervalu rangova.
ZINCRBY	Uvećava rang elementa u uređenom skupu za zadati inkrement.
ZRANGE	Vraća elemente uređenog skupa u zadatom intervalu pozicija. Ukoliko je zadat opcioni parametar *WITHSCORES,* naredba vraća elemente sa njihovim rangovima.
ZRANK	Vraća poziciju elementa u uređenom skupu.
ZSCORE	Vraća rang elementa u uređenom skupu.

Tabela 4. *Naredbe za rad sa Redis uredjenim skupom*

Redis heš

Heš u Redisu predstavlja kolekciju atribut - vrednost parova. I atribut i vrednost u Redis hešu su tipa String pa je heš zapravo preslikavanje elemenata tipa String u elemente tipa String. Stoga, za njih važe osobine koje važe za Redis string - mogu se interpetirati na razne načine u zavisnosti od toga šta sadrži. Heševi su veoma pogodni za grupisanje elemenata koji semantički pripadaju zajedno, pa se često koriste za skladištenje kompleksnih objekata. Na primer, za objekat tipa auto, heš atributi bi mogli biti “boja”, “snaga motora” itd.

Redis heš se može porediti sa redom u relacionoj bazi ili sa dokumentom u dokument-orijentisanoj bazi, i moguće je baratati pojedinačnim ili višestrukim atributima odjednom.

Tipične operacije za rad sa heševima date su u tabeli 5.

Ime naredbe	Opis naredbe
HSET	Postavlja atribut na zadatu vrednost.
HGET	Vraća vrednost atributa.
HGETALL	Vraća sve atribute i njihove vrednosti.
HDEL	Briše atribute iz heša.

Tabela 5. *Naredbe za rad sa Redis hešom*

Mapa bitova

Mapa bitova nije novi tip podatka, već je posebna vrsta niski koja sadrži sekvencu nula i jedinica. Indeksi bitova u mapi bitova nazivaju se često pozicijama i koriste se na različite načine u zavisnosti od aplikacije. Nule se posmatraju kao “isključeni” bitovi, dok su jedinice “uključeni“ bitovi.

Mape bitova su efikasne za analitiku u realnom vremenu - da li se neki događaj desio, da li je korisnik izvršio određenu akciju u određenom vremenskom intervalu ili broj poseta veb sajtu određenog datuma.

Memorijska efikasnost Mape bitova se može uvideti poređenjem sa skupom, na primeru broja poseta sajtu određenog datuma. Poziciju u Mapi bitova predstavlja identifikator korisnika. Pretpostavimo da naša aplikacija ima 5 miliona korisnika, a da je na zadati datum koristilo 2 miliona korisnika, i da se svaki identifikator predstavlja uz pomoć 4 bajta. Kod Redis skupa potrebno je čuvati identifikator svakog korisnika koji je posetio sajt kao poseban element skupa. Kod Mape bitova potrebno je čuvati jedinice na onim pozicijama jednakim identifikatoru korisnika koji su posetili sajt određenog datuma. U najgorem slučaju u implementaciji pomoću Mape bitova, moraće se alocirati memorija za čitav opseg

vrednosti polja ID, ukoliko je korisnik sa najvišom vrednošću polja ID (ID od 5000000) posetio sajt tog datuma. Međutim, Mapa bitova nije efikasna kada je broj posetilaca mnogo manji od broja korisnika, na primer 100, i tada će ona opet iskoristiti 5 miliona bitova u najgorem slučaju (tabela 6).

Postoje posebne naredbe koje operišu njima i dele se u dve grupe: operacija nad pojedinačnim bitovima i operacija nad grupom bitova.

Broj posetilaca sajta	Struktura podatka	Broj bitova po korisniku	Broj identifikatora korisnika koje čuvamo	Iskorišćena memorija
2 miliona	Mapa bitova	1	5 miliona	1b x 5000000 = 625kB
2 miliona	Skup	32	2 miliona	32b x 2000000 = 8MB
100	Mapa bitova	1	5 miliona	1b x 5000000 = 625kB
100	Skup	32	100	32b x 100 = 0.4kB

Tabela 6. *Primer poredjenja Mape bitova i Skupa*

Ime naredbe	Opis naredbe
SETBIT	Postavlja vrednost bita na zadatoj poziciji.
GETBIT	Vraća vrednost bita na zadatoj poziciji.
BITOP	Izvršava bitovsku operaciju između zadatih Mapa bitova.
BITPOS	Vraća poziciju prvog bita postavljenog na zadati bit (0 ili 1).

Tabela 7. *Naredbe za rad sa Mapom bitova*

HyperLogLog

HyperLogLog je struktura podataka koja se koristi za procenu broja jedinstvenih elemenata unutar zadate liste vrednosti. Koristi probabilistički algoritam koji procenjuje kardinalnost skupa različitih elemenata sa greškom od oko 1%. Veoma je koristan kada potpuna preciznost nije jedan od zahteva pri rešavanju problema, dok ušteda vremena i prostora jeste. Za deterministički algoritam brojanja jedinstvenih elemenata potrebno je izdvojiti onoliko memorije koliko vrednosti treba izbrojati, zato što je potrebno zapamtiti elemente koji se ponavljaju, kako se ne bi brojali više puta. Kod HyperLogLog strukture podataka potrebno je rezervisati 12kB po ključu u najgorem slučaju. Kodirani su kao Redis string, ali postoje posebne naredbe za njih - za dodavanje, brojanje i spajanje stringova. Česti slučajevi korišćenja:

- Brojanje različlitih upita izvršenih od strane korisnika;
- Brojanje različitih tagova koje korisnik prati;
- Brojanje različitih reči koji se pojavljaju u knjizi.

Komande za rad sa strukturom podataka HyperLogLog su date u tabeli 8.

Ime naredbe	Opis naredbe
PFADD	Dodaje elemente u HyperLogLog.
PFCOUNT	Vraća približnu vrednost jedinstvenih elemenata unutar HyperLogLog vrednosti.
PFMERGE	Pripaja HyperLogLog vrednosti.

Tabela 8. *Naredbe za rad sa HyperLogLog tipom podataka*

Postojanost podataka

Jedna od glavnih karakteristika koja Redis izdvaja u odnosu na druge baze podataka je to što podatke čuva u RAM memoriji. Pristup RAM memoriji je znatno brži od pristupa sekundarnoj memoriji, pa su zato i operacije nad podacima znatno brže.
RAM memorija je nestalna i svi podaci bi nestali ukoliko bi server morao ponovo da se pokrene, ukoliko dođe do pada sistema i slično. Redis nudi dva načina trajnog čuvanja podataka u sekundarnoj memoriji: pomoću kreiranja i čuvanja „snimka“ baze na disk (engl. snapshotting) i pomoću upisivanja svih izvršenih naredbi pisanja u posebnu datoteku na disku (engl. append-only file, AOF). Može se koristiti jedan mehanizam, kombinacija oba ili nijedan od njih, zavisno od slučaja korišćenja.

-Snimak baze-

Snimkom baze u Redisu pravi se datoteka koji predstavlja podatke smeštene u bazi do tog trenutka. Postoji više načina izvođenja snimka baze:

- Pomoću naredbe SAVE koja je blokirajuća naredba i stopira sve upite ka Redisu dok se snimanje ne završi.

- Pomoću naredbe BGSAVE (background save) koja ne zaustavlja rad servera prilikom snimanja podataka. Ona kreira dete proces (fork) koji se bavi snimanjem podataka, dok roditelj proces sve vreme prima naredbe od klijenata i izvršava ih.

Oblik naredbe SAVE: *SAVE broj_sekundi broj_promena*

Naredba SAVE, sa argumentima broj_sekundi i broj_promena, kaže da će se čuvanje trenutnog stanja baze pokrenuti automatski na svakih broj_sekundi ukoliko se u tom intervalu desio broj_promena operacija pisanja od poslednje uspešno izvršene naredbe SAVE. U datoteci koja

sadrži podešavanja Redisa može biti navedeno više takvih naredbi istovremeno, i čim se prva od njih ispuni, pokreće se proces pisanja datoteke.

Koliko god često podešavali iniciranje čuvanja snimka skladišta, nije zagarantovano da neće biti izgubljenih podataka. Ukoliko dođe do stopiranja Redis servera, sve promene nad postojećim, i svi novokreirani podaci nakon poslednjeg snimka će biti izgubljeni. Još jedna mana ovog mehanizma je što prilikom BGSAVE naredbe kreiranje procesa deteta može za cenu imati potpuno stopiranje sistema ukoliko imamo posla sa velikom količinom podataka koja ne ostavlja puno memorije za dodatno kreiranje procesa deteta. U tom slučaju, zbog bitke za resurse, sâm proces čuvanja snimka trenutnog stanja Redis skladišta traje duže, i dolazi do značajnog pogoršavanja performansi sistema. Kako bi se to izbeglo, može se koristiti varijanta sa SAVE blokirajućom naredbom, ili se može potpuno isključiti automatsko iniciranje snimka i pokretati se ručno onda kada je najmanje frekventno korišćenje baze.

-AOF-

AOF predstavlja mehanizam upisivanja u datoteku svake od naredbi pisanja u bazu, redosledom kojim se izvršavaju. Time se kompletna baza može rekreirati tako što se izvrše naredbe iz datoteke. Prilikom pisanja na disk, podaci se prvo upisuju u tzv. bafer, a operativni sistem zatim u nekom trenutku uzima podatke i upisuje ih na disk. Pošto postoji vremenski razmak između te dve operacije, postoji i šansa da dođe do gubljenja podataka ukoliko u međuvremenu dođe do gašenja servera. Zato postoje opcije sinhronizacije koje podešavaju koliko često će se vršiti sinhronizacija bafera i diska: prilikom svake naredbe, svake sekunde ili bez podešavanja (operativni sistem bi na svoj način vršio sinhronizaciju).

Najsigurnija opcija vršenja sinhronizacije je prilikom svake naredbe, ali je ona najsporija i zavisi od performansi diska (npr. za SSD je kobno često upisivanje malog broja informacija). Optimalna opcija koja daje kompromis onoga što nudi i onoga što odnosi je opcija sinhronizovanja

bafera i diska svake sekunde – ne opterećuje se disk, a u slučaju pada servera su izgubljeni podaci pristigli u toku jedne sekunde.

Međutim, AOF datoteka može značajno narasti, s obzirom da se u nju dodaje na kraj svaka izvršena naredba. Takođe, prilikom ponovnog pokretanja Redis servera, rekreiranje podataka iz datoteke AOF može potrajati ukoliko je datoteka dugačka. To se rešava podešavanjem opcije za prepisivanje datoteke u drugu, spajajući više srodnih naredbi u jednu. Tu se javlja isti problem kao kod snimka baze i kreiranja procesa deteta koji radi prepisivanje.

Replikacija Redisa

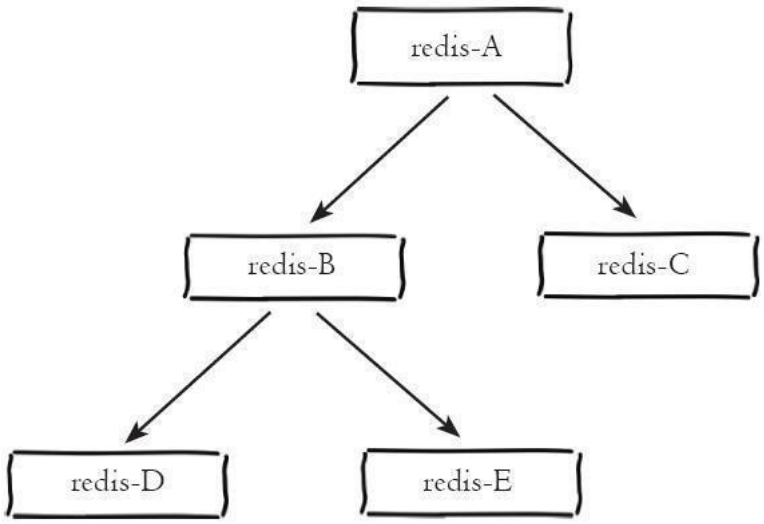
Replikacija baze podataka je metoda kojom se postiže njeno skladištenje na više lokacija. Arhitektura sistema u kom se koristi replikacija Redisa predstavlja arhitekturu glavni/podređeni (engl. master/slave). Služi u svrhe skalabilnosti, tako što sve operacije čitanja obavljaju replike, dok master obavlja operacije pisanja.

Kada se podređeni Redis server poveže na glavni Redis server, glavni server započinje BGSAVE operaciju. Kada završi, počinje sa slanjem snimka baze podređenoj instanci Redis servera. Podređena instanca tada briše sve podatke koje je sadržala i počinje sa učitavanjem pristiglog snimka baze. Ukoliko je u intervalu od startovanja BGSAVE naredbe, do primanja snimka baze na replici, bilo nekih operacija pisanja, glavna instanca ih čuva u datoteci i šalje podređenoj instanci, koja ih kasnije izvršava. Kao rezlultat, glavna i podređena instanca servera sadrže iste podatke.

Nakon što podređeni server primi prvi put snimak baze podataka, on se održava ažurnim pri svakom pisanju na glavni server. Klijent koji ima potrebu za čitanjem, povezuje se na podređeni server, i tako smanjuje opterećenje na glavnom serveru. Opterećenje glavnog servera se može

umanjiti i isključivanjem opcije za trajno čuvanje podataka, kako ne bi trošio vreme na ulazno/izlazne operacije.

Kada su zahtevi za čitanjem znatno veći od zahteva za pisanjem, preporučuje se dodavanje novih podređenih Redis instanci, čime se smanjuje opterećenje postojećih servera. Međutim, tako se može doći u situaciju gde jedan glavni server ne može da izvršava brzo pisanja na podređene servere. Tada se preporučuje ubacivanje novog sloja čvorova glavni/podređeni, koji pomaže u podeli dužnosti pri replikaciji (slika 2).



Slika 2. *Primer drvolike replikacije Redis servera*

Replikacija, u kombinaciji sa AOF metodom trajnog čuvanja podataka, se pokazala kao stabilan model otporan na pad sistema i gubljenje podataka.

Topologija koja je inicijalno dizajnirana za Redis sistem sastojala se iz Redis instanci u kom glavna instanca servera prima sve naredbe pisanja i replicira ih na podređene instance servera. Takva topologija radi dobro u slučajevima kada:

- Master ima dovoljno memorije da skladišti sve podatke;
- Prihvatljivo je zaustaviti izvršavanje aplikacije koja koristi Redis, kada je potrebno zaustaviti glavni Redis server.

Ali ne i u slučajevima kada:

- Količina podataka premašuje memoriju dostupnu na glavnom serveru;
- Izvršavanje aplikacije koja koristi Redis ne sme biti zaustavljeno;
- Potrebno je distribuirati podatke;
- Nije prihvatljivo otkazivanje čitavog sistema usled otkazivanja Redis servera.

Da bi odgovorili na takve izazove, razvijeni su projekti Redis Klaster i Redis Sentinel. Redis Klaster ima za cilj distribuiranje podataka i automatsko rešavanje problema sa otkazivanjem rada glavnog servera. Redis Sentinel ima za cilj da pruži automatsko rešavanje problema otkazivanja glavnog servera u glavno/podređenom sistemu Redis instanci.

Zaključak

Redis skladište je različito od ostalih baza podataka u mnogo segmenata - koristi primarnu memoriju servera kao primarno skladište, a disk za trajno čuvanje podataka, radi u jednoj niti sa stanovišta izvršavanja naredbi, operacije pristupa i ažuriranja podataka su atomične, model podataka je jedinstven. Podržava kompleksne strukture podataka koje predstavljaju vrednost, dok je ključ uvek tipa string. S druge strane, svi podaci moraju stati u memoriju, nema mogućnosti kreiranja upita i podacima se uvek pristupa preko jedinstvenog ključa (skladište se ne može pretraživati po vrednosti).

Redis skladište se dobro pokazalo u raznim primenama u softverskom inženjerstvu. Može služiti u svrhe kreiranja redova zadataka, u svrhe održavanja tabele s rezultatima i rangiranja, publish/subscribe mehanizma, keširanja stranica, sesije i podataka. Redis takođe može služiti kao primarna baza podataka, s ograničenjem da svi podaci moraju stati u primarnu memoriju. U poređenju sa svojim konkurentima na tržištu, Redis daje dobre rezultate u terminima brzine pristupa podacima. U poređenju sa keš tehnologijama, on ima podršku perzistencije podataka i kompleksnije strukture podataka kojima se lakše i prirodnije modeluju podaci.

Pokazalo se da je Redis veoma koristan alat u terminima performansi softvera. Osim toga, veoma je jednostavno Redis platformu uvrsititi u ekosistem postojeće aplikacije. Tome svedoče i brojni primeri reinženjeringa velikih veb aplikacija, poput Instagrama, Twittera, StackOverflow i druge.

Literatura

[1] Victor Zakhary, Divyakant Agrawal, Amr El Abbadi, Caching at the Web Scale, SantaBarbara, California, 2017, https://cs.ucsb.edu/~victorzakhary/assets/papers/caching-at-theweb-scale.pdf

[2] Redis dokumentacija, https://redis.io/

[3] Josiah L. Carlson, Redis In Action, Manning Publications Co, Shelter Island, New York, 2013. ISBN: 9781617290855

[4] Hugo Lopes Tavares, Maxwell Dayvson Da Silva, Redis Essentials, Packt Publishing, Birmingham, United Kingdom, 2015. ISBN: 9781784392451

[5] RedisLabs white papers, 15 reasons to use Redis as an Application Cache,

https://redislabs.com/docs/15-reasons-caching-is-best-done-with-redis/