NoSQL baza podataka računalnih komponenti

Anić, Vladimir

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
ELEKTROTEHNIČKI FAKULTET

Sveučilišni studij

NOSQL BAZA PODATAKA RAČUNALNIH KOMPONENTI

Završni rad

Vladimir Anić

Osijek, 2016

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 01.09.2016.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada

Ime i prezime studenta:	Vladimir Anić
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R3533, 30.07.2013.
OIB studenta:	20443345723
Mentor:	Doc.dr.sc. Ivica Lukić
Sumentor:	
Naslov završnog rada:	NoSQL baza podataka računalnih komponenti
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Predložena ocjena završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 Postignuti rezultati u odnosu na složenost zadatka: 3 Jasnoća pismenog izražavanja: 3 Razina samostalnosti: 3
Datum prijedloga ocjene mentora:	01.09.2016.
Datum potvrde ocjene Odbora:	12.09.2016.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA O ORIGINALNOSTI RADA

Osijek, 13.09.2016.

Ime i prezime studenta:	Vladimir Anić
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R3533, 30.07.2013.
Ephorus podudaranje [%]:	0

Ovom izjavom izjavljujem da je rad pod nazivom**: NoSQL baza podataka računalnih komponenti**

izrađen pod vodstvom mentora Doc.dr.sc. Ivica Lukić

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA

Ja, Vladimir Anić, OIB: 20443345723, student/ica na studiju: Preddiplomski sveučilišni studij Računarstvo, dajem suglasnost Elektrotehničkom fakultetu Osijek da pohrani i javno objavi moj **završni rad**:

NoSQL baza podataka računalnih komponenti

u javno dostupnom fakultetskom, sveučilišnom i nacionalnom repozitoriju.

Osijek, 13.09.2016. 2016.

_____

potpis

SADRŽAJ

1. UVOD

NoSQL baze podataka su baze koje pružaju mehanizme za pohranu i dohvaćanje podataka. Za razliku od klasičnih relacijskih baza podataka NoSQL baze imaju mogućnosti rada sa bazama podataka puno većih razmjera u puno manjem vremenu.

MongoDB izbjegava klasični oblik relacijskih baza podataka i zamjenjuje ih dinamičnim shemama i korištenjem formata koji se naziva BSON. Na taj način čini integraciju podataka u određene tipove aplikacija jednostavnije i brže.

Tema završnog rada je izrada NoSQL baze podataka računalnih komponenti. Zadatak je izvršen koristeći MongoDB softversko rješenje. Osim same baze podataka također je potrebno dati teoretsku osnovu NoSQL baza podataka, a ponajviše MongoDB bazu, te je potrebno napraviti usporedbu sa relacijskim bazama i pokazati prednosti i nedostatke relacijskih, odnosno NoSQL baza podataka.

U drugom poglavlju rada napravljene su važne teorijske osnove bitne za razumijevanje NoSQL baza podataka, odnosno objašnjena je teorija o NoSQL bazama podataka, više koncentrirano na dokument baze podataka i sam MongoDB.

U trećem poglavlju napravljeni su i objašnjeni MongoDB model baze računalnih komponenti i relacijski model računalnih komponenti kako bi se mogle usporediti razlike.

Konačno, u četvrtom poglavlju napravljena je usporedba MongoDB i relacijskog modela na primjeru i općenita usporedba MongoDB i relacijskih baza podataka.

1.1. Zadatak završnog rada

Zadatak završnog rada je modelirati NoSQL bazu računalnih komponenti koristeći MongoDB bazu podataka. Također je potrebno dati teoretsku osnovu NoSQL baza podataka i konkretnije MongoDB baza podataka te navesti prednosti i nedostatke NoSQL baza podataka u odnosu na relacijske baze podataka.

2. VAŽNE TEORIJSKE OSNOVE

2.1. NoSQL baze podataka

NoSQL okružja baza podataka su ne-relacijski i široko distribuirani sustavi koji omogućuju brzu organizaciju i analizu visoko volumnih, različitih tipova podataka. Ono podrazumijeva rješenja koja se ne zasnivaju na relacijskim sustavima upravljanja baza podataka i većinom su otvorenog koda, horizontalno skalabilne, jednostavne, BASE, te rade sa velikim količinama podataka. Strukture podataka korištene od strane NoSQL baza podataka su drugačije od onih korištenih u relacijskim bazama podataka što ih čini bržima od relacijskih baza podataka u pojedinim slučajevima. Za NoSQL baze podataka vrijedi CAP (eng. Consistency, eng. Availability, eng. Partition tolerance) teorem i to ponajviše za toleriranje particija (eng. Partition tolerance) – uvjeti u kojima su neke skupine računala izolirane. Osim toleriranja particija još je važna i dostupnost (eng. Availability) – uvijek odgovaranje na zahtjeve. Većina NoSQL baza podataka nudi „eventualnu dosljednost“ u kojoj se promjene baze podataka propagiraju na sve čvorove „na kraju“ odnosno najčešće unutar nekoliko milisekundi. Zbog toga se upiti za određene podatke neće možda odmah vratiti sa ažuriranim podacima i to može dovesti do čitanja podataka koji u tom trenutku nisu točni. Ovaj problem je poznat kao ustajalo čitanje. Kada se kaže da su NoSQL baze BASE to znači da su one uvijek dostupne unatoč kvarovima (eng. Basically Available), da se sustav stalno mijenja pa ne moraju uvijek biti dosljedne (eng. Soft-state) i da će se sve izmjene u konačnici provesti i svima biti dostupne (eng. Eventual consistency).

NoSQL baze se mogu podijeliti prema modelu podataka na:

- Dokumente – koriste ključ kako bi pomoću njega dobili dokument. Neke od ovih baza podataka su: MongoDB, CouchDB, SimpleDB…
- Ključ–vrijednosti – koriste ključ na temelju kojeg dobivaju vrijednosti. Ove baze uključuju: Cassandra , MUMPS, DynamoDB…
- Spremanja u stupac – umjesto spremanja podataka u redove, spremaju podatke u stupce. Najpoznatije su: BigTable, HyperTable, HBase…
- Grafovi – koriste čvorove i bridove za spremanje podataka. Primjeri ovih baza su: GraphDB, InfoGrid, Neo4j…
- Objektno-orijentirane baze podataka – prihvaća semantiku objekata podržanu u objektno-orijentiranom programiranju. Ovdje spadaju: JADE, ObjectDB, ObjectDatabase++…

2.2. Dokument baze podataka

2.2.1. Općenito

Dokument baze podataka su računalni programi stvoreni za spremanje, dohvaćanje i upravljanje informacijama koje su orijentirane k dokumentima, odnosno još se nazivaju i polustrukturirani podaci. Ove baze podataka su jedne od glavnih NoSQL baza podataka i ovakva baza bit će korištena u ovome radu. Dokument orijentirane baze podataka su inherentni podrazred ključ-vrijednost spremanja podataka koji je drugi koncept NoSQL baza podataka. Dokument baze podataka se znatno razlikuju od relacijskih baza podataka. Relacijske baze podataka spremaju podatke u posebne tablice koje je napravio i definirao programer i jedan objekt se može koristiti u više tablica, dok kod dokument baza podataka to nije slučaj. Dokument baze podataka spremaju sve informacije za dani objekt u jednu instancu baze podataka i svaki spremljeni objekt može biti drugačiji od drugog. Ovo omogućuje mapiranje objekata u bazi podataka jednostavnim i otklanja sve slično objektno-relacijskom mapiranju, što ih čini poželjnim za programiranje web aplikacija koje su podvrgnute stalnom mijenjanju mjesta i gdje je brzina implementacije važna za rad.

2.2.2. Dokumenti

Glavni koncept dokument baza podataka su sami dokumenti. Dok se svaka implementacija dokument baze podataka zasniva na različitim dokumentima, može se ipak reći da je definicija ista – koriste zadani ključ i pomoću njega dobivaju dokument. Ključevi su jedinstveni ključevi koji predstavljaju taj dokument. Ključ je jednostavni identifikator (ID), najčešće niz znakova, URI ili staza. Dokumenti baze podataka enkapsuliraju i kodiraju podatke u standardnim formatima ili kodovima, a najčešći su: JSON, BSON, XML, YAML, neki drugi polustrukturirani format, te binarni podaci. Može se reći da su dokumenti kod dokument baza podataka isto što i koncept objekata u objektno-orijentiranom programiranju. Oni se ne moraju držati neke standardne sheme, niti moraju imati iste dijelove, ključeve, odjeljke itd.

Osnovne operacije sa dokument bazama podataka:

- Unos (ključeva, dokumenata)
- Dohvat (ključeva)
- Ažuriranje (ključeva, dokumenata)
- Brisanje (ključeva)
- Dohvat na temelju sadržaja dokumenta – nije standardno jer nema upitnog jezika

2.3. MongoDB

MongoDB je baza podataka otvorenog tipa osnovana na modelu dokument baza podataka. MongoDB ne koristi klasične tablice kao relacijske baze podataka, već je osnovan na binarnim JSON dokumentima (BSON) sa dinamičnim shemama. Ovaj način integriranja podataka u određenim situacijama čini upravljanje podacima bržima i lakšima. Baza podataka je izdana 2009. besplatna i otvorenog tipa pod dvojnom eng. licencom (GNU i Apache licence).

2.3.1. Modeliranje MongoDB baze podataka

Podaci u MongoDB bazama podatak imaju fleksibilnu shemu, a dokumenti u istim kolekcijama ne trebaju imati iste setove polja ili struktura i svi slični dijelovi u kolekciji dokumenata mogu sadržavati različite tipove podataka.

Kada se dizajnira shema u MongoDB-u moramo uzeti u obzir neke posebnosti, a to su:

- Dizajn sheme ovisi o zahtjevima korisnika
- Kombiniranje objekata u jednom dokumentu ako će se koristiti zajedno, inače ih treba odvojiti
- Dupliciranje podataka jer je diskovni prostor jeftin, za razliku od vremena računanja
- Spajanje treba vršiti pri upisivanju, a ne čitanju podataka
- Optimiziranje sheme za najčešće slučajeve korištenja
- Vršenje složenih agregacija u shemi

2.3.2. Glavne značajke

Eng. Ad hoc upiti – MongoDB podržava polja, upite raspona te pretraživanje normalnih izraza. Upiti mogu povratiti određena polja ili dokumente, te mogu uključivati i korisnički definirane JavaScript funkcije. Također, mogu se namjestiti da vraćaju nasumične uzorke rezultata korisnički zadane veličine.

Indeksiranje – indeksi podržavaju učinkovito rješavanje upita. Bez indeksa MongoDB mora pretražiti svaki dokument u kolekciji kako bi izabrao onaj dokument koji se nalazi u upitnoj izjavi. Ovo pretraživanje je neučinkovito i zahtjeva od mongod-a (MongoDB poslužitelj) da obradi ogromne količine podataka. Zbog tog razloga postoje indeksi koji su specijalne strukture podataka koje spremaju mali dio podataka napravljen tako da se može lakše pretražiti. Indeks sprema vrijednosti određenog polja ili seta polja poslaganog po vrijednosti polja određenog u indeksu.

Repliciranje – proces sinkronizacije podataka koristeći veći broj servera. Repliciranje omogućuje redundanciju i povećava dostupnost podataka sa višestrukim kopijama na različitim poslužiteljima baza podataka. Repliciranje također dozvoljava vraćanje podataka ukoliko dođe do kvara u sklopovlju i prekida usluge, odnosno sa višestrukim kopijama podataka moguće je posvetiti jedan poslužitelj kritičnim situacijama.

Balansiranje tereta – MongoDB skalira horizontalno koristeći se dijeljenjem na krhotine (eng. sharding). Korisnik izabire ključ krhotine koji određuje kako će podatci u kolekciji biti distribuirani. Podatci su razdvojeni u raspone (osnovane na ključu krhotine) i distribuirani na više krhotina. Krhotina je glavna (eng. master) sa jednim ili više podređenih (eng. slaves). Kao što je već spomenuto, MongoDB može raditi na više poslužitelja i tako balansirati teret i/ili replicirati podatke kako bi podatci bili dostupni i u slučaju kvara hardvera. MongoDB je jednostavan za razvitak i novi uređaji se mogu uvijek dodati već postojećim bazama podataka.

Sustav spremanja podataka – MongoDB može biti i korišten kao datotečni sustav koristeći se prednostima balansiranja podatak i replikacije podataka na više uređaja za spremanje datoteka i podataka. Ovo se naziva povezani datotečni sustav i uključen je s MongoDB pogonskim programima, te je dostupan za mnoge razvojne jezike. U sustavima sa više uređaja MongoDB-a datoteke mogu biti distribuirane i kopirane više puta između uređaja i s time učinkovito stvaraju sustav sa balansiranim teretom i mogućnosti pogreške.

Agregacija – smanjenje mape se može koristiti za procesiranje hrpa i agregatne operacije. Okvir agregacije omogućuje korisniku da dobije rezultate koje bi u SQL-u dobili naredbom GROUP BY. Operatori agregacije se mogu spojiti kako bi formirali cjevovod – analogan Unix cijevima. Okvir agregacije uključuje operator $lookup koji može pridružiti dokumente iz više različitih dokumenata, kao i statističke operatore kao što je standardna devijacija.

Izvršenje JavaScript-a na strani poslužitelja – JavaScript može biti korišten u upitima, agregatnim funkcijama ili može biti izravno poslan bazi podataka na izvršenje.

2.3.3. Arhitektura

Dostupnost programskih jezika – MongoDB ima službene pogonske programe za razne popularne programske jezike i razvojna okruženja. Također postoji veći broj neslužbenih ili pogonskih programa razvijenih od zajednice za ostale programske jezike i okvire.

Upravljanje i grafički sustavi – većina administracije se radi pomoću alata linije naredbi kao što je mongo ljuska jer MongoDB ne uključuje nikakvo administrativno grafičko sučelje, no postoje proizvodi i projekti ostalih proizvođača koji pružaju korisničko sučelje za

administraciju i pregled podataka. Jedno od ovih sučelja je i Robomongo koji je korišten za ovaj projekt.

Licenciranje i podrška – MongoDB je dostupan pod besplatnom eng. GNU Affero General Public License licencom. Jezični pogonski programi su dostupni pod eng. Apache licencom, a sam MongoDB Inc. daje licencu za korištenje za MongoDB.

3. MODELIRANJE SUSTAVA

3.1.MongoDB model računalnih komponenti

Baza se sastoji od sedam glavnih računalnih komponenti koji su stvoreni kao kolekcije i mogu se gledati kao ekvivalenti tablicama u relacijskom modelu baza podataka.

Kolekcija „procesor“ – omogućuje uvid u glavne atribute računalnih procesora, odnosno možemo vidjeti tko je proizvođač, model procesora, njegov takt i broj jezgri, zatim priključak za matičnu ploču, njegovu ugrađenu predmemoriju i njegov ID koji samo služi kako bi lakše pronašli procesor. ID ne mora biti primarni ključ jer MongoDB sam dodjeljuje i svoj _id kad se stvori objekt u kolekciji i na taj način rješava problem identificiranja objekata u bazi podataka.

Kolekcija „graficka_kartica“ – omogućuje uvid u glavne atribute grafičkih kartica računala. Oni su: proizvođač čipa, proizvođač, model, VRAM, brzina procesora, port, potrebno napajanje i ID.

Kolekcija napajanje – sadrži glavne atribute napajanja računala. Atributi kolekcije napajanje su: proizvođač, snaga, tip, korisnost i ID.

Kolekcija „memorija“ – u svojim objektima sadrži glavne atribute radne memorije računala, a to uključuje: proizvođač, kapacitet, takt, tip i ID.

Kolekcija „kucista“ – omogućuje uvid u glavne atribute koji su potrebni za izbor kućišta računala, a to su: proizvođač, tip, dimenzije i ID.

Kolekcija „hdd“ – sadrži glavne atribute potrebne za izbor odgovarajućeg tvrdog diska za računalo. Glavni atributi objekata su: proizvođač, kapacitet, RPM i ID.

Kolekcija „maticna_ploca“ – se sastoji od objekata koji imaju opise svega što je potrebno za izbor matične ploče, a to su: proizvođač, priključci, chipset i ID.

Kako bi se uopće mogle napraviti kolekcije potrebno je napraviti lokalni poslužitelj baze podataka na koji se može spojiti i započeti sa stvaranjem baze i kolekcija. Lokalni poslužitelj se može stvoriti nakon što se instalira MongoDB programski paket i nakon toga u eng. command prompt-u je potrebno aktivirati, tj. napraviti server naredbom mongod u datoteci u kojoj je instalirano MongoDB programsko okruženje. Nakon toga se moguće spojiti na server i započeti pisati bazu podataka. Baza podataka se najlakše kreira koristeći naredbu „use DATABASE_NAME“. U ovom primjeru baza podataka se zove „Racunalne_komponente“ i sintaksu za stvaranje baze moguće je vidjeti u priloženome kodu (Prilog [1]). Nakon što je stvorena baza podataka mogu se stvoriti i kolekcije u bazi. Za stvaranje kolekcija koristi se

naredba „db.createCollection(name, options)“ gdje u „name“ pišemo ime kolekcije, a u „options“ dodajemo dokument koji određuje opcije za veličinu memorije i indeksiranje. „Options“ je neobavezan dodatak pri stvaranju kolekcije i kao takav nije korišten. U ovome primjeru stvoreno je sedam entiteta, kao što je gore i navedeno te u prilogu je vidljivo kako se stvaraju sve kolekcije. Za unos podataka i atributa u kolekcije koristimo naredbu „db.COLLECTION_NAME.insert({document})“, gdje je COLLECTION_NAME ime stvorene kolekcije, a u zagradu upisujemo dokumente, odnosno pridružujemo vrijednosti atributima. U kodu (P.3.1.) se može vidjeti da su dokumenti uneseni ulančano u kolekcije koristeći uglate zagrade ([]), kao polje u C programskom jeziku.

Za bazu podataka računalnih komponenti je bitna, osim atributa svih glavnih komponenata, brzina pretraživanja. Za smanjivanje broja dokumenata koji se pretražuju i time povećavanja efikasnosti baze koristi se indeksiranje. Indeksi u ovoj bazi moraju biti postavljeni na atribute koji su najčešće pretraživani i koji će najviše pomoći u smanjenju vremena potraživanja. Jedna od stvari na koju se mora obratiti posebna pozornost pri indeksiranju je postavljanje samih indeksa na te atribute. Ukoliko bi indeksi bili postavljeni na svaki atribut vrijeme traženja bi se znatno smanjilo, ali samo dok ne bi morali ažurirati tablicu. Pri ažuriranju tablice je potrebno i ažurirati svaki pojedini indeks, a time se posao administratora produljuje i zato za efikasan rezultat se indeksi postavljaju, kao što je već napomenuto, samo na atribute koji se najviše pretražuju.

Indeksi za kolekciju „procesor“ su postavljeni na atribute koji su najčešće pretraživani, a to su proizvođač i broj jezgri u procesoru.

Indeksi za kolekciju „graficka_kartica“ stavljeni su na atribute koje potrošači najčešće pretražuju, a ti atributi su proizvođač čipa i model grafičke kartice.

Indeksi za kolekciju „napajanje“ se nalaze na atributu proizvođač i atributu snaga jer korisnici najčešće prvo traže snagu napajanja, a zatim pouzdanog proizvođača.

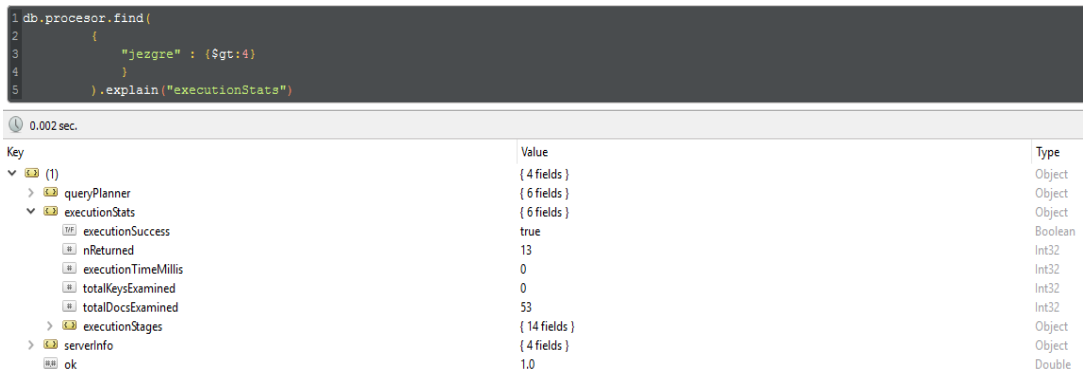
Indeksi za kolekciju „memorija“ su postavljeni na atributima proizvođač i kapacitet jer su to dvije stvari koje se najčešće pretražuju prilikom kupnje.

Indeksi za kolekciju „kucista“ su stavljeni na atribut proizvođača jer određeni potrošači preferiraju određene proizvođače i na atribut tip jer je to podatak koji se najviše pretražuje prilikom izbora kućišta.

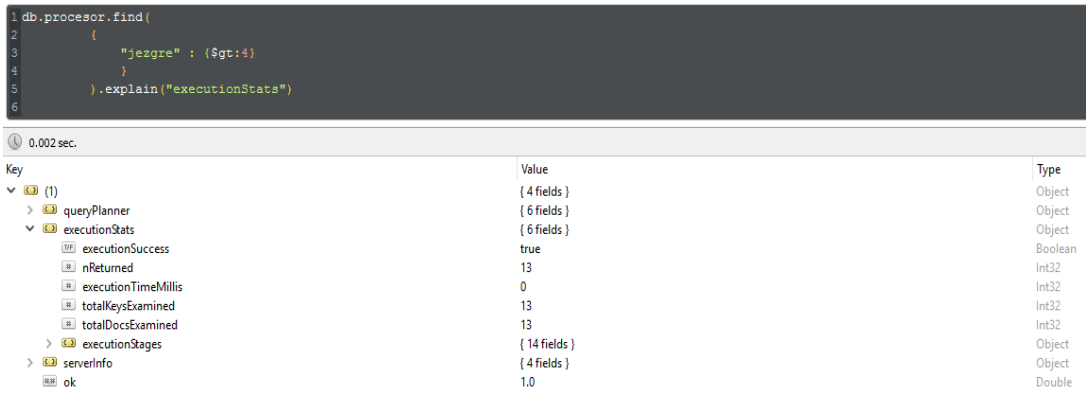
Indeksi za kolekciju „hdd“ se nalaze na atributu kapacitet i atributu RPM jer su to najvažnije i najviše pretražene karakteristike prilikom odabira tvrdih diskova.

Indeksi za kolekciju „maticna_ploca“ su postavljeni na atribute proizvođač i chipset jer prilikom odabira matične ploče ljudi najčešće traže pouzdane proizvođače i određeni chipset koji određen modelom izabranog procesora.

Na primjeru indeksa za kolekciju procesor postavljenog na atribut jezgre možemo vidjeti znatno poboljšanje u pretraživanju. Na slici 3.1. možemo vidjeti da korištenjem funkcije eng. find za pronalazak svih procesora koji imaju više od četiri jezgre funkcija pretražuje svaki pojedini dokument (eng. totalDocsExamined na slici 3.1.) i mora proći kroz sva 53 procesora unesena u bazu, kako bi vratila 13 procesora koji odgovaraju rezultatu pretraživanja (eng. nReturned na slici 3.1.). Na slici 3.2. je vidljivo znatno poboljšanje pri ovom pretraživanju. Kao što se može vidjeti kada se koriste indeksi za pronalazak, pregledavaju se samo oni dokumenti koji odgovaraju pretraživanju i u velikim bazama podataka to znatno smanjuje vrijeme pretraživanja i omogućuje izravan pristup dokumentima.



Sl. 3.1. Pretraživanje procesora u bazi sa brojem jezgri većim od 4 bez korištenja indeksa



Sl. 3.2. Pretraživanje procesora u bazi sa brojem jezgri većim od 4 uz korištenje indeksa

Korištene su agregatne funkcije su ovoj bazi podataka kako bi se pojedine kolekcije grupirale po određenom atributu, a zatim nad njima obavljale različite operacije. Ovo je bitno zato što je pojedine podatke potrebno i statistički bilježiti, te računati i pronalaziti pojedine

vrijednosti svih dokumenata u kolekciji. Prilikom korištenja agregatnih funkcija uvijek je potrebno grupirati dokumente po nekom atributu (to je zapravo sama agregacija atributa) i tek onda se mogu koristiti sve ostale funkcije.

Agregatne funkcije za kolekciju „procesor“ su grupirane po atributu priključak za matičnu, a tražen je skup svih priključaka za matičnu ploču. Osim skupa svih priključaka za matičnu ploču također je napisana agregatna funkcija koja prema priključku za matičnu ploču traži prosječnu brzinu procesora i vraća nam vrijednosti po grupi kao što je prikazano na slici 3.3.

Agregatna funkcija za kolekciju „hdd“ grupira sve tvrde diskove prema atributu proizvođač i vraća ukupni broj tvrdih diskova po pojedinom proizvođaču.

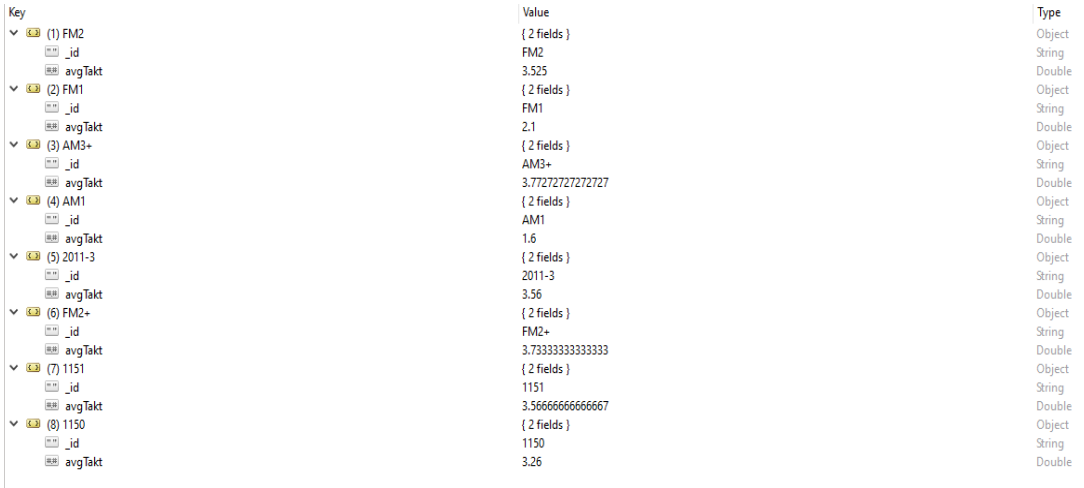
Agregatna funkcija za kolekciju „maticna_ploca“ grupira sve dostupne matične ploče u bazi prema atributu chipset i vraća ukupan broj ploča prema pojedinom chipsetu.

Agregatna funkcija kolekcije „kucista“ grupira sva kućišta dostupna u bazi po njegovom tipu i vraća nam ukupan broj kućišta po njemu samome.

Agregatna funkcija kolekcije „memorija“ grupira sve radne memorije prema njihovom tipu i vraća takt memorije po pojedinom tipu.

Agregatna funkcija kolekcije „napajanje“ grupira sva napajanja prema atributu tip i za rezultat vraća prosječnu korisnost napajanja po tipu.

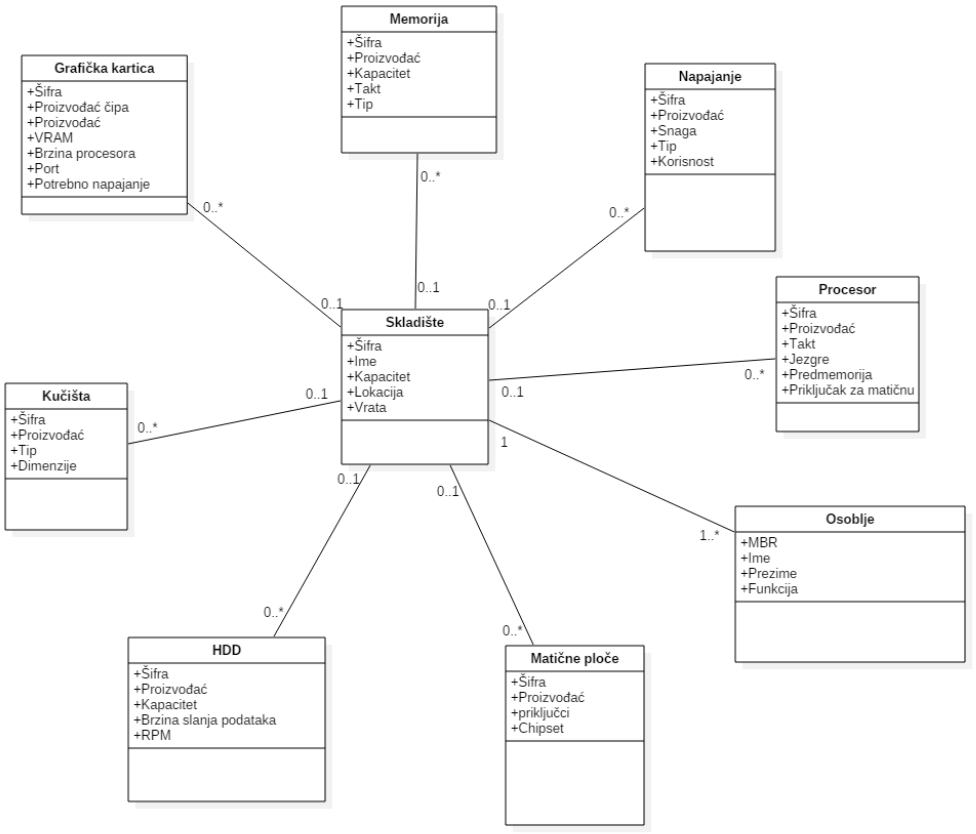
Agregatna funkcija kolekcije „graficka_kartica“ grupira sve grafičke kartice u bazi prema njihovom modelu, a za rezultat vraća prosjek potrebnog napajanja po samome modelu.



Sl. 3.3. Prosječna brzina procesora ovisno o vrsti chipseta

3.2. SQL model računalnih komponenti

Za SQL model računalnih komponenti korišteni su prethodno napravljen ER (eng. Entity-Relationship) dijagram i model. Na slici 3.4. vidljiv je ER dijagram za skladište računalnih komponenti. U ovom konkretnom slučaju je potrebno skladište da se mogu povezati entiteti SQL modela kao što sam upitni jezik nalaže.



Sl. 3.4. ER (eng. Entity-Relationship) dijagram baze podataka računalnih komponenti

Na prikazanome dijagramu možemo vidjeti da su svi entiteti povezani sa tablicom skladište. Iz veza s kojima su povezani entiteti možemo vidjeti da je Šifra Skladišta (primarni ključ tablice Skladište) strani ključ u svim ostalim entitetima. Ovo se najbolje vidi iz tablice 3.1. gdje ŠifraS predstavlja šifru skladišta u ER dijagramu.

Tab. 3.1. Veze entiteta i ključeva

Entitet	**Primarni ključ**	**Strani ključ**
Skladište	Šifra	
Osoblje	Šifra	ŠifraS
Procesor	Šifra	ŠifraS

Napajanje	Šifra	ŠifraS
Memorija	Šifra	ŠifraS
Grafička kartica	Šifra	ŠifraS
Kučišta	Šifra	ŠifraS
HDD	Šifra	ŠifraS
Matične ploče	Šifra	ŠifraS

Primjenu i definiranje stranih ključeva može se vidjeti u Prilogu [2]. Kao što se može uočiti kako bi napravili strani ključ potrebno je napisati ime ključa na koji se odnosi, tj. entitet od kojeg uzimamo primarni ključ kako bi postao strani i da prilikom brisanja glavne tablice se briše i strani ključ u navedenoj tablici. U Prilogu [2] se također može vidjeti način unosa podataka u tablice korištenjem naredbe „INSERT INTO Table_Name VALUES ()“.

4. USPOREDBA MODELA NOSQL I RELACIJSKE BAZE PODATAKA

Za usporedbu modela korištene su već spomenute baze podataka računalnih komponenti. Model relacijske baze podataka moguće je vidjeti na slici 3.4. i ovdje počinje jedna od prvih razlika između dva modela bazi podataka. Za razliku od relacijskih baza podataka koje se zasnivaju na jednome od ER modela (Chenov model, James Martinov – IE model, IDEF1X ili UML model), NoSQL baze imaju dinamičnu shemu koja nije unaprijed određena i moguće ju je mijenjati. Ovakav pristup je dobar za velike baze koje su pogodne promjenama jer je moguće lakše snalaženje po bazi i ažuriranje podataka. Ovo je također dobro ako dođe do brisanja kolekcija, jer oni nisu izravno povezani već se mogu gledati kao zasebne cjeline. Na konkretnom primjeru modela, razlika između dinamične i statičke sheme se vidi po korištenju entiteta skladište i s time posljedično dolazi i do stvaranja entiteta zaposlenici jer su potrebni zaposlenici koji rade u skladištu. Bez entiteta skladište nije bilo moguće napraviti relacijsku bazu podataka, tj. povezati računalne komponente u jednu bazu kao što je to napravljeno pomoću MongoDB dokument baze podataka.

Jedna od možda i najbitnijih razlika između relacijskih i NoSQL baza je to što kod NoSQL baza nije potrebno ručno dodjeljivanje primarnog ključa. MongoDB automatski stvara primarni ključ u obliku 12-bitnog hexadecimalnog broja („_id: ObjectId(broj)“). Strani ključ također nije potreban zato jer veze između kolekcija ne zahtijevaju poveznicu, već se pišu kao ulančani dokumenti (dokumenti unutar dokumenta).

Korištenjem veza kod relacijskih baza podataka smanjena je mogućnost duplikacije podataka kako ne bi došlo do miješanja atributa istog imena. Također ovo se smatra prvom normalnom formom u SQL-u, tj. svi ne ključni atributi moraju biti potpuno ovisni o ključu. Kod MongoDB ako se dogodi duplikacija podataka, tj. duplikacija dokumenta MongoDB će ih kategorizirati kao dva različita dokumenta i svakome od njih pridružiti vlastiti _id.

Mnogi koncepti koji se primjenjuju u SQL-u imaju analogiju u MongoDB-u. U tablici 4.1. vidimo te analogije, odnosno koncepte koji su česti u oba jezika, a smatraju se ravnopravnima.

Tab. 4.1. Analogija relacijske baze podataka i MongoDB-a

SQL	**MongoDB**
Tablica	Kolekcija
Red	Dokument
Stupac	Polje/atribut

Veza	Ulančani dokument

MongoDB i SQL oboje imaju bogat izbor upita koje koriste i mogu se povući neke paralele između oba jezika. U tablici 4.2. imamo upite u kojima se vidi razlika između SQL i MongoDB upita. Jedna od najbitnijih razlika je da pri unošenju podataka u tablicu kod relacijskih baza definiramo sve atribute prije samog unošenja u tablicu dok se kod MongoDB-a stvaranjem kolekcije ne određuje koji su atributi u samoj kolekciji, već se to radi prilikom unošenja.

Tab. 4.2. Razlika upita kod relacijske i MongoDB baze podataka

	SQL	**MongoDB**
Kreiranje tablice/kolekcije	CREATE TABLE Procesor(Sifra INT PRIMARY KEY, Proizvodac VARCHAR(30), Takt VARCHAR(10) DEFAULT '2.8 GHz', Jezgre INT DEFAULT 2, Predmemorija VARCHAR(10) DEFAULT '4 MB', Priključak za matičnu VARCHAR(20) DEFAULT 'AM 3+', SifraS INT, CONSTRAINT fk_Skladiste FOREIGN KEY(SifraS) REFERENCES Skladiste(Sifra) ON DELETE CASCADE)	db.createCollection("procesor")

Unos podataka	INSERT INTO Procesor VALUES
	('6759', 'Intel', '3.2 GHz', '4', '4 MB', '3350'),
	("0264', 'AMD', '4.8 GHz', '8', '8 MB', 'AM 3+'),
	('6725', 'AMD', '3.8 GHz', '6', '4 MB', 'FM 2');
		db.procesor.insert(
		[
		{
		"proizvodac":"AMD",
		"model":"Phenom II x4 955",
		"id":1,
		"takt":3.2,
		"jezgre":4,
		"predmemorija":"4 MB",
		"prikljucak za maticnu":"AM3+",
		},NAKON ZAREZA SE MOGU UNOSITI ULANČANO OSTALI DOKUMENTI
		]
		}
		)

Dohvaćanje podataka	SELECT * FROM Procesor	db.procesor.find()

Grupiranje podataka	SELECT Priključak za matičnu, COUNT(*) FROM Procesor GROUP BY Priključak za matičnu	db.procesor.aggregate(
		{
		$group : {
		_id : "$prikljucak za maticnu",
		total : {$sum : 1}
		}
		}
		)

Prilikom dohvaćanja podataka u bazi, naredbe u SQL-u i MongoDB-u se ne razlikuju previše osim same sintakse. Također moguće je naprednije dohvaćanje podataka, ali u tablici 4.2. prikazan je samo osnovni primjer dohvaćanja podataka.

Za grupiranje podataka u MongoDB jeziku se koriste agregatne funkcije. Agregatne funkcije su proces u kojemu se grupiraju vrijednosti iz više dokumenata zajedno te se nad tom grupom podataka mogu obavljati razne operacije koje će vratiti jedan rezultat. U relacijskom modelu podataka možemo reći da je upit SELECT, uparen sa upitima COUNT(*) i GROUP BY se može reći da odgovaraju agregatnim funkcijama u MongoDB-u.

Indeksiranje je bitan dio MongoDB-a jer ono znatno skraćuje broj pretraženih dokumenata, a s time i vrijeme pretraživanja kao što se može vidjeti na slikama 3.1. i 3.2. U relacijskim bazama podataka također postoje indeksi i služe za brže pretraživanje. Kreiranjem indeksa nad relacijom znači stvaranje binarnog stabla nad željenim atributom. Kod MongoDB-a kreiranje indeksa se također vrši nad atributom u kolekciji, a može se shvatiti kao telefonski imenik, odnosno indeksiranjem u MongoDB-u slažemo dokumente u kolekciji redom po traženome atributu kao što su imena složena u telefonskom imeniku. Na taj način prilikom pretraživanja računalo, odnosno MongoDB, zna točno gdje tražiti zadani kriterij te neće trošiti vrijeme i resurse na pretraživanje svih dokumenata, nego će proći samo kroz one koji sadrže tražene podatke. Važno je naglasiti da se indeksi i kod relacijskih i kod MongoDB baza podataka trebaju staviti na atribute koji se često koriste u pretragama. Indeksi se mogu staviti na sve atribute u bazi podataka, ali time će se smanjiti njihova korisnost. Kod relacijskog modela se indeksi ne bi trebali stvarati ako relacija sadrži mali broj n-torki jer u tom slučaju binarno stablo ne pridonosi smanjenju vremena pretrage.

MongoDB se koristi za aplikacije velikih razmjera zato što je dinamične sheme i može se lako mijenjati, te nije predodređen kao relacijske baze podataka. Zbog toga što je dokument baza podataka, podacima se može izravno pristupiti te korištenjem indeksa nad podacima koji se najviše pretražuju, smanjeno je i samo vrijeme pretraživanja. Repliciranje u MongoDB-u omogućuje da se dijelovi baze podataka spremaju na više različitih poslužitelja što u relacijskim bazama podataka nije moguće te se MongoDB baza replicira kako bi stalno ostala raditi prilikom prekida rada jednog od poslužitelja (ako se uopće koristi više poslužitelja). Relacijske baze podataka su pogodnije za sustave kojima su potrebne manje baze podataka ili složene(više-redne) transakcije jer MongoDB trenutno ne podržava složene transakcije. U priloženoj bazi podataka neke prednosti koje koristi MongoDB nisu mogle biti pokazane zbog nedostatka resursa i korištenja jednostavnog lokalnog poslužitelja (repliciranje i balansiranje tereta), no

pokazani su osnovni aspekti korištenja i razlike između relacijskog i MongoDB modela baze podataka.

5. ZAKLJUČAK

Cilj ovog rada bio je napraviti NoSQL bazu računalnih komponenti koristeći MongoDB i usporediti ga sa relacijskim modelom baze računalnih komponenti. Također je bilo potrebno napraviti teorijsku osnovu NoSQL baza podataka.

Zadatak je uspješno napravljen te se u prilogu nalazi MongoDB baza računalnih komponenti i relacijska baza računalnih komponenti korištena za usporedbu. Napravljena baza nije dovoljno velika kako bi se vidjela velika razlika u brzini izvođenja, ali pokazani su osnovni principi i sintaksa za pisanje MongoDB baze podataka i usporedba sa relacijskom bazom podataka. Osim same baze podataka također su određeni indeksi i agregatne funkcije u MongoDB bazi podataka, te osnovne operacije sa relacijskom bazom podataka.

LITERATURA

[1] http://www.planetcassandra.org/what-is-nosql/ - NoSQL općenito, 15.06.2016.

[2] http://nosql-database.org/ - NoSQL općenito, 16.06.2016.

[3] I. Lukić, Distribuirane baze podataka, Baze podataka predavanje 11, Loomen stranica kolegija Baze podataka, 19.06.2016.

[4] http://mdslab.unime.it/sites/default/files/mongodb_tutorial.pdf - upute za korištenje MongoDB-a, 15.07.2016.-15.08.2016.

[5] https://www.mongodb.com/compare/mongodb-mysql - usporedba MongoDB-a i SQL-a, 21.08.2016.

[6] I. Lukić, Fizička organizacija baze podataka, Baze podataka predavanje 10, Loomen stranica kolegija Baze podataka, 22.08.2016.

[7] I. Lukić, Distribuirane baze podataka, Baze podataka predavanje 11, Loomen stranica kolegija Baze podataka, 23.08.2016.

SAŽETAK

NoSQL baza računalnih komponenti

Ovaj rad sadrži teorijsku osnovu NoSQL baza, MongoDB bazu računalnih komponenti, te usporedbu relacijskih i NoSQL baza podataka. U prilogu se nalazi NoSQL i relacijska baza, te neke opcije koje se mogu koristiti prilikom pisanja i korištenja NoSQL i relacijskih baza podataka.

Ključne riječi: NoSQL, MongoDB, dokument baze podataka, relacijski model

ABSTRACT

Title: NoSQL database for computer components

This paper contains theoretical background for NoSQL databases, MongoDB database for computer components and comparison of relational and NoSQL databases. Apendix contains NoSQL and relational database with some options that can be used for writing and using NoSQL and relational databases.

Keywords: NoSQL, MongoDB, document oriented databases, relational model

ŽIVOTOPIS

Vladimir Anić rođen je 09.03.1995. godine u Osijeku. Od rođenja živi u Belišću gdje stječe osnovnoškolsko obrazovanje u Osnovnoj školi Ivana Kukuljevića Belišće. Godine 2009. upisao se u Srednju školu Valpovo smjer Elektrotehnika. Godine 2013. završava srednju školu i državnu maturu, te upisuje Elektrotehnički fakultet, preddiplomski studij računarstva, u Osijeku. Godine 2016 sudjeluje na Elektrijadi u Riminiju iz kolegija Engleski jezik gdje ekipno postiže drugo mjesto, no zbog tehničke prirode poništen je taj dio natjecanja.

PRILOZI

[1] Kod.txt – kod NoSQL baze računalnih komponenti u .txt formatu

[2] SQL kod.txt – kod relacijske baze podataka u .txt formatu

[3] Zavrsni_rad.js – MongoDB baza računalnih komponenti