NoSQL dokument baza podataka: prikaz skladištenja podataka sa osvrtom na podatke sa senzora

Olivera Janković
"ORAO" a.d.
Bijeljina, BiH
janolja@yahoo.com

Sadržaj — **Prilikom odabira i projektovanja odgovarajućih modela podataka, potrebno je pored ostalog uzeti u obzir svojstvenu strukturu samog podataka. Senzorski podaci su u tom smislu doista specifični, traže modele u kojima će priroda tih podataka biti predstavljena na najbolji mogući način, posebno u kontekstu potrebe njihove integracije sa ostalim podacima. U radu će biti predstavljan koncept skladištenja podataka dokumentima orijentisan, te jedan način skladištenja senzorskih podataka zasnovan na dokumentima - korištenjem NoSQL dokument baze podataka MongoDB.**

Ključne riječi –NoSQL dokument baze podataka; MongoDB; podaci vremenskih serija; senzorski podaci;

I. UVOD

Dug period mnogih profesionalnih karijera u oblasti razvoja softvera, relacione baze podataka su bile podrazumjevani izbor, profesionalan pristup problemu smještanja podataka. Pitanje je u stvari bilo za koju bazu podataka, odnosno za kojeg proizvođača se odlučiti. Svojim dugotrajnim i dokazanim postojanjem postale su važna karika, moglo bi se reći kultni dio računarskog razmišljanja. U tom kontekstu pojava koncepta NoSQL (Not Only SQL) baza podataka dolazi kao svojevrstan vid iznenađenja.

Važni aspekti vezani za domen NoSQL baza podataka su još uvijek u fazi turbulencije, mjenjaju se u vremenu nudeći nova rješenja, nove mogućnosti i nove baze podataka. Međutim, pored nesporne činjenice „nezrelosti“ i stoga upitne dugovječnosti nekih ponuđenih rješenja evidentno je da sama ideja i koncept NoSQL zaslužuje pažnju, nudi nešto novo jer je nastao iz realnih izazova i naraslih potreba za novim načinom poimanja i predstavljanja modela podataka. NoSQL baze podataka mogu se posmatrati kao potencijalno rješenje, pomak na putu uklanjanju problema tradicionalnog načina projektovanja i korištenja relacionih baza podataka [1], [2]. Činjenica je da današnje poslovanje crpi snagu iz podataka nastalih u sve kraćim i frekventnijim vremenskim intervalima, što neminovno vodi ka velikim količinama podataka. Koncept je u mnogome i nastao iz potrebe da se pored ostalog, na adekvatan način, zadovolje zahtjevi obrade velike količine podataka.

Relacione baze podataka u svom osnovnom formatu nisu dizajnirane za smještanje i upravljanje podacima vremenskih serija (*time-series*). Samo korištenje podataka vremenskih serija za poslovne analize u svojoj biti nije novi pokret. Ono što je novo, je mogućnost za prikupljanje i analizu ogromne količine podataka vremenskih serija, iznimno velikom brzinom kako bi dobili najjasniju sliku u kontekstu predikcije. Cilj je dakle predvidjeti buduće tržišne promjene, ponašanja korisnika, uslova životne sredine i okruženja, korištenja resursa, zdravstvene trendove i još mnogo, mnogo više.

U ovom radu će biti prikazana konceptualna osnova NoSQL baza podataka, sa naglaskom na rješenja zasnovana na dokumentima, kroz testne primjere korištenja reprezentativnog predstavnika dokument baza podataka MongoDB. Dodatan naglasak dat je na mogućnost i opciju predstavljanja podataka vremenskih serija, kroz ilustrativan primjer predstavljanja i skladištenja podataka sa senzora.

II. NOSQL BAZE PODATAKA

NoSQL obuhvata široku paletu tehnologija baza podataka a nastao je i razvio se kao odgovor na sveprisutan, značajan porast obima podataka iz raznih tipova izvora, frekvenciju kojom se pristupa ovim podacima, zahtjev na performanse i potrebe obrade. Osnovni tipovi NoSQL baza podataka su:

- Baze podataka zasnovane na dokumentima (*Document databases*),
- Baze podataka zasnovane na (orijentisane ka) grafovima (*Graph database*),
- Baze podataka - skladišta podataka iz grupe ključ-vrijednost (*Key-value stores*),
- Baze podataka - kolonski orijentisano skladištenje podataka (*Wide-column stores*).

A. *Neke prednosti NoSQL baza podataka*

U poređenju sa relacionim bazama podataka NoSQL baze podataka su skalabilnije i nude bolje performanse a njihovi modeli podataka daju rješenja na neka od pitanja kojima relacioni modeli nisu dizajnirani da se bave, kao što su i:

- velike količine struktuiranih, polustruktuiranih i nestruktuiranih podataka,

- agilne metode razvoja[1], brz i agilan razvoj aplikacija,
- objektno-orijentisano programiranje koje je fleksibilno i jednostavnije za učenje,
- efikasna proširiva (*scale-out*) arhitektura umjesto skupe monolitne arhitekture.

B. Pitanje šeme

Kao što je poznato, relacione baze podataka zahtjevaju definisanje šeme baze podataka. Ovo se uklapa u pristup nedovoljno agilnog razvoja jer svaki put kada se uoče neke nove osobine realnog sistema šema baze podataka treba da se promjeni. Tako, ako je potrebno da se npr. kupcu proširi postojeći skup atributa, tj. da se dodaju nove stavke (npr. omiljena boja, vrsta robe, ...) pored uobičajenih (identifikacioni podaci kupca, adresa, telefon,...) neophodno je prvo dodati novu kolonu a zatim migrirati cijelu bazu podataka na novu šemu. Ako je pri tome baza podataka i „dovoljno“ velika ovo može biti i spor proces koji posljedično može da dovede i do značajnog zastoja u radu. Takođe, nema načina da se u slučaju korištenja relacionih baza podataka efikasno koriste nestruktuirani podaci ili podaci čija struktura nije unaprijed poznata.

S druge strane, NoSQL baze podataka su kreirane da dozvole umetanje podataka bez unaprijed definisane šeme. Time je značajno olakšana realizacija promjene aplikacije u realnom vremenu, bez brige o situacijama prekida, što znači da je razvoj brži, kod integracije je pouzdaniji i nema vremena odvojenog za administratora baze podataka.

C. Distribuirano smještanje podataka

Zbog načina na koji su struktuirane relacione baze podataka kada je u pitanju hardverska podrška uobičajena je vertikalna skalabilnost, tj. obično zahtjevaju da jedan server bude domaćin cijele baze podataka da bi se osigurala pouzdanost i kontinuirana dostupnost bazi podataka. Na povećane potrebe se u principu odgovara dogradnjom serverskih kapaciteta (brži procesor, veći hard diskovi,...) što vremenom postaje skupo. Solucija je horizontalna skalabilnost – dodavanje novih servera umjesto povećanja kapaciteta jednog servera.

Distribuirano smještanje podataka, ili kako je uobičajeni termin u NoSQL svijetu tzv. *sharding* baze podataka na više instanci servera može biti postignuto SQL bazama podataka ali se u praksi obično ostvaruje u okviru mreža za spremanje podataka (SAN, Storage Area Network) ili drugim kompleksnim aranžmanima koji omogućavaju da se složen hardver ponaša kao jedan server. Budući da relacione baze podataka ne obezbjeđuju nativnu podršku (nije u skladu sa njihovom „prirodom“) razvojni timovi su zaduženi za implementaciju više relacionih baza podataka na određeni broj mašina. Podaci se pohranjuju u svakoj instanci baze podataka autonomno. Aplikacioni kod se razvija da distribuira podatke, distribuira upite, kao i za objedinjavanje rezultantnih podataka preko svih instanci baze podataka. Pored toga, potrebno je razviti dodatni kod za obradu kvarova i grešaka resursa, za realizaciju spajanja (*join*) širom različitih baza podataka, za rebalansiranje, replikaciju i druge zahtjeve. Takođe, mnoge prednosti u relacionim bazama podataka kao što je transakcijski integritet su ugroženi ili se eliminišu u slučaju ručnog particionisanja.

NoSQL baze podataka sa druge strane obično podržavaju automatsko particionisanje što znače da nativno i automatski šire podatke preko proizvoljnog broja servera bez potrebe da aplikacija vodi računa o kompoziciji serverskog skupa (*server pool*). Podaci i učitavanje upita su po automatizmu izbalansirani širom servera, a u slučaju pada servera on može biti transparentno zamjenjen bez prekida aplikacije

Pored toga, računarstvo u oblaku (*cloud computing*) je znatno lakše a usluge kao što su Amazon Web Service pruža virtuelan, gotovo neogračen kapacitet na zahtjev i pri tome vodi brigu o svim neophodnim poslovima administracije baze podataka [3]. Oni koji se bave razvojem aplikacija više ne moraju graditi složene, skupe platforme kako bi podržali svoje aplikacije – već se mogu koncentrisati na pisanje aplikacionog koda. Pristupačni serveri (*Commodity Server*) mogu obezbjediti iste mogućnosti obrade i skladištenja kao jedan server visoke klase (*high-end server*) za znatno nižu cijenu.

Mnoge NoSQL baze podataka također podržavaju automatsku replikaciju što u konačnom daje visoku dostupnost i oporavak od katastrofe bez potrebe uvođenja novih posebnih aplikacija za upravljanje zadacima toga tipa. Okruženje za smještanje podataka značajno je virtuelizovano iz perspektive razvojnog programera.

III. NoSQL DOKUMENT BP - MongoDB

MongoDB [2] sistem za upravljanje bazama podataka otvorenog koda (ver. 2.6.5) je među vodećim NoSQL bazama podataka, razvijen od strane firme “10gen”, oktobra 2007. god. [4].

MongoDB smješta sve podatke u dokumente, koristeći pri tome JSON[3] (**J**ava**S**cript **O**bject **N**otation)) stil, struktura podataka koja je sastavljena od parova polje&vrijednost (Sl.1).

```
{"naziv": "Zafira", "model": "A", "proizvodnja": "1996-2006"}
```

Slika 1. Primjer JSON formata

Svaki dokument se sastoji od jednog ili više atributa čija vrijednost odgovara tipu podataka JSON formata (broj, string, Boolean, niz,...). MongoDB skladišti dokumente u BSON (binarni JSON) formatu koji je vrlo sličan JSON formatu.

Ono što je najbitnije, svi podaci su vezani na dokument čime se uveliko smanjuje vrijeme pristupa podacima i gotovo u potpunosti izbacuje potreba za kompleksnim spajanjem tabela. Naravno, u ovom slučaju potencijalno sporno pitanje je pitanje ažuriranja podataka. Međutim najčešće se entiteti koji su

[1] Iterativne i postupne metodologije za razvoj softvera – princip je dat kroz “Manifesto for Agile Software Development” (Beck i dr., 2001)

[2] Na spisku imena korisnika MongoDB su i: MetLife, Forbes, Bosh, Expedia, City of Chicago,...

[3] Tekstualni standard za razmjenu podataka, lako čitljiv za čovjeka, nezavisan niti o jednom programskom jeziku

podložni promjenama ipak ne spremaju direktno na dokument, već se pohranuju zasebno a dokument ima spremljenu asocijaciju na taj entitet.

MongoDB dakle spada u već pomenutu grupu, baze podataka zasnovane na dokumentima (document databases), koje skladište podatke u obliku dokumenata, za razliku od relacionih baza podataka koje podatke skladište unutar tabela. U tabeli I su predstavljeni uporedni nazivi, terminologija relacionih i NoSQL baza podataka. Kolekcija (*Collection*) je grupa dokumenata koji imaju neka zajednička svojstva i može se reći da odgovara pojmu tabele u konceptu relacionih baza podataka.

TABELA I. UPOREDNI NAZIVI KOD RELACIONIH I NOSQL BAZA PODATAKA

RDBMS	MongoDB
Tabela	Kolekcija (*Collection*)
Red	JSON Dokument
Index	Index
JOIN[4]	Referenca ili ugrađeni dokument

U NoSQL bazama podataka, za razliku od SQL baza podataka šema je fleksibilna, odnosno posmatrano iz ovog konteksta kolekcija ne zahtijeva tačnu strukturu dokumenata koji se u nju smještaju. Naravno, dokumenti koji se u nju pohranjuju, tj. dokumenti koji se pohranjuju u okviru iste kolekcije u biti imaju sličnu strukturu jer predstavljaju slične entitete. Ova fleksibilnost olakšava mapiranje dokumenta u neki entitet ili objekat.

Važna odluka kod dizajniranja modela podataka je izbor načina na koji će dokumenti biti povezani jedni sa drugima. Postoje dva načina povezivanja i to su reference i ugrađeni dokumenti (JOIN klauzula kod RDBMS, Tabela I). Reference čuvaju odnos između dokumenata korištenjem linkova ili referenci iz jednog dokumenta u drugi (tzv. normalizovani model podataka), dok se ugrađivanjem dokumenata veze između podataka spremaju unutar jednog dokumenta.

```
{
_id: "ISBN: 978-86-521-1551-8",
Naslov: "Gavrilov princip",
Autor: "Grupa autora",
Godina: 2014,
Strana: 320,
Izdavac: "Laguna",
Zanr: [("roman"), ("istorijski roman")]
}
```

Slika 2. Prikaz koda zapisa knjiga

Kao pokazni primjer uzet je slučaj neke knjižare u kojoj se za svaku knjigu prate određeni podaci. Jedan način predstavljanja u formi dokumenta, pokazan na primjeru jedne knjige, prikazan je na Sl. 2. Na Sl. 3 je prikazan način kreiranja dokumenta na primjeru dokumenta „knjiga“ i njegovo smještanje u kolekciju nazvanu „knjige“, korištenjem mongo shell-a (interaktivni JavaScript shell interfejs za MongoDB). Svaki MongoDB dokument, ukoliko se ne uradi drugačije, po automatizmu[5] dobiva jedinstven identifikator id. U ovom slučaju, kao što se može vidjeti sa Sl. 2 i Sl. 3 vrijednosti id je direktno pridružen međunarodni standardni knjižni broj ISBN, kao „prirodan“ identifikator svake knjige.

```
>
>
> knjiga = { _id: "ISBN: 978-86-521-1551-8", Naslov: "Gavrilov princip", Autor: "Grupa autora",
Godina: 2014, Strana: 320, Izdavac: "Laguna", Zanr: [("roman"), ("istorijski roman")] }
{
    "_id" : "ISBN: 978-86-521-1551-8",
    "Naslov" : "Gavrilov princip",
    "Autor" : "Grupa autora",
    "Godina" : 2014,
    "Strana" : 320,
    "Izdavac" : "Laguna",
    "Zanr" : [
        "roman",
        "istorijski roman"
    ]
}
> db.knjige.save(knjiga)
WriteResult({ "nMatched" : 0, "nUpserted" : 1, "nModified" : 0, "_id": ISBN: 978-86-521-1551-8 })
>
```

Slika 3. Prikaz kreiranja dokumenta ”knjiga” i njegovo smještanje u kolekciju ”knjige” korištenjem mongo shell-a

Koristeći komandu db.knjige.find(), vrši se upit nad kolekcijom „knjige“ kao što se može vidjeti na Sl. 4 dok komanda „show collections“ (koja joj prethodi na Sl. 4) vraća nazive kolekcija. (Sam upit vraća prvih dvadeset dokumenata, za ostale se kuca „it“ u okviru mongo shell-a.)

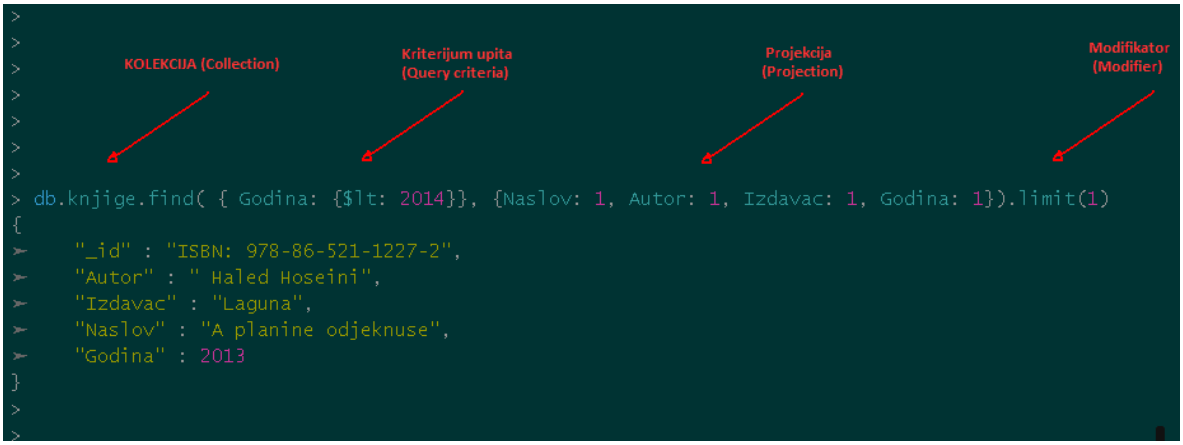
```
> show collections
knjige
> db.knjige.find()
{
    "_id" : "ISBN: 978-86-521-1687-4",
    "Strana" : 272,
    "Godina" : 2014,
    "Autor" : " Paulo Kueljo",
    "Naslov" : "Preljuba",
    "Izdavac" : "Laguna",
    "Zanr" : [
        "roman",
        "ljubavni roman"
    ]
}
{
    "_id" : "ISBN: 978-86-521-1551-8",
    "Strana" : 320,
    "Godina" : 2014,
```

Slika 4. Upit nad kolekcijom ”knjige” korištenjem mongo shell-a

[4] SQL klauzula za spajanje tabela

[5] Mongo shell obezbeđuje klasu ObjectId() koja generiše novi ID objekta

Prethodni upit bi odgovarao upitu „select * from knjige“ kod relacionih baza podataka, koji vraća sve redove tabele knjige. Primjer realizacije kompleksnijih upita koji sadrže kriterijume upita, projekcije i/ili modifikatore prikazan je na Sl.5. Rezultat pretrage (Sl.5) u ovom primjeru treba da je jedna knjiga (limit(1)), koja je izdata prije 2014. godine ($lt: 2014) pri čemu su traženi atributi : Naslov, Autor, Izdavac, Godina (Naslov: 1, Autor: 1, Izdavac: 1, Godina: 1).



Slika 5. Prikaz primjera upita sa zadatim kriterijumom upita

IV. SMJESTANJE PODATAKA TIPA VREMENSKIH SERIJA

Podaci koji dolaze sa izvora podataka u vremenskim sekvencama, kao sa trake, obično predstavljaju određene opservacije, rezultate praćenja ili posmatranja poredane u vremenskom redoslijedu [5]. Takvih izvora je veoma mnoga, a neki od njih su:

- Finansijska tržišta koja generišu cijene i vrijednost širokog asortimana proizvoda, akcija i sl..
- Razna transportna sredstva koja proizvode podatke tipa: lokacija, brzina,...
- Stalna ažuriranja statusa na socijalnim mrežama.
- Pozivi, SMS i drugi signali koji dolaze sa mobilnih uređaja.
- Razne vrste sistema koji tokom svoga rada neprekidno bilježe podatke i kreiraju tzv. logove podataka.
- Senzori koji zavisno od namjene mjere razne ambijentalne parametre tipa temperatura, pritisak, vlažnost i ostalo.

Svim pomenutim izvorima su dakle zajedničke karakteristike sekvence (serije) podataka u vremenu koje je nadalje neophodno predstaviti kroz adekvatne modele podatake u cilju njihove iskorištenosti a u kontekstu očuvanja semantike podataka.

A. Model za podatke vremenske serije – podaci sa senzora

Smještanje podataka sa senzora u relacione baze podataka u osnovi se može predstaviti kao pohranjivanje svakog očitanja podataka sa senzora (npr. vrijeme očitanja, vrijednost očitanja) u tabelu. Recimo da se prati temperatura nekog sistema/prostora svake minute. (Informacija sa temperaturnih senzora, ukoliko se temperatura ne mijenja naglo, prenose se obično u većim vremenskim intervalima). U tom slučaju parovi očitanja mogu se predstaviti kao primjer pokazan u tabeli II. u kojoj su za potrebe ilustracije prikazane vrijednosti tri uzastopna očitanja temperature (sa vremenskim razmakom od 1 minute).

TABELA II. PRIKAZ SENZORSKIH PODATAKA U TABELI (RDBMS)

Vrijeme_ocitanja	Ocitana temperatura
2014-11-10 10:00:00	12.35
2014-11-10 10:01:00	12.00
2014-11-10 10:02:00	11.45

Ako bi se direktno mapirao ovaj pristup smještanja podataka u MongoDB dobili bi jedan dokument po događaju (Sl. 6)

```
{
 "Vrijeme_ocitanja": "2014-11-10 10:00:00",
 "tip": "Ocitana temperatura",
 "vrijednost": "12.35"
 },
{
 "Vrijeme_ocitanja": "2014-11-10 10:01:00",
 "tip": "Ocitana temperatura",
 "vrijednost": "12.00"
},
{
 "Vrijeme_ocitanja": "2014-11-10 10:02:00",
 "tip": "Ocitana temperatura",
 "vrijednost": "11.45"
 }
```

Slika 6. Primjer u kome je svako očitanje temperature predstavljeno jednim dokumentom

Drugi pristup modeliranja polazi od toga da se višestruka očitanja smjeste u jednom dokumentu (Sl. 7), na primjer jedan dokument po satu u kome je i dalje očuvana rezolucija podataka od jedne minute. Kako bi se poboljšala efikasnost, u

šemi se mogu izolovati ponavljajuće strukture podataka. U polju „vrijeme očitanja sat“ se može spremiti sat koji semantički određuje dokument (za ovako napisan dokument MongoDB bi po automatizmu generisao već pomenuti jedinstven identifikator dokumenta id), a za svako očitanje temperature čuva se nova vrijednost u poddokumentu „vrijednosti“, obzirom da se smještanje jedne vrijednosti temperature u minuti može jednostavno predstaviti kao polja od 00÷59.

```
{
   "Vrijeme_ocitanja_sat": "2014-11-10
10:00:00",
   "tip": "Ocitana temperatura",
   "vrijednosti" : {
         "00": 12.35,
         "01": 12.00,
         "02": 11.45,
          ---

         "59": 11.50
  }
}
```

Slika 7. Višestruka očitanja smještena u okviru jednog dokumenta (rezolucija 1 minut)

U slučaju da postoji potreba da su očitanja sa senzora veće rezolucije npr. svake sekunde tada bi se moglo predstaviti na sličan način kao na Sl. 7, pri čemu bi se polje „vrijednosti“ umjesto od „00“ do „59“, logično kretalo od „00“ do 3599“. Prednosti ovoga pristupa je da se vrijednostima jednog sata očitanja može pristupiti jednim čitanjem dokumenta, za razliku od pristupa prikazanog na Sl. 6 (svako očitanje novi dokument) gdje je potrebno čitanje 3600 dokumenata (3600 ako je rezolucija 1 sekunda, odnosno 60 za rezoluciju 1 minut) što u krajnjem rezultira znatno boljim performansama. Međutim ovaj pristup ima i značajno lošu stranu, npr. da bi se ažurirala zadnja sekunda sata neophodno je proći cijelu dužinu vrijednosti (3600 koraka). Poboljšana efikasnost može se ostvariti kroz drugi način zapisa za smještanje očitanja sa senzora sa rezolucijom od jedne sekunde, u okviru jednog dokumenta, a koji je prikazan na Sl. 8.

```
{
  "Vrijeme_ocitanja_sat": "2014-11-10
10:00:00",
  "tip": " Ocitana temperatura",
  "vrijednosti" : {
     0: { 0:12.35, 1:12.35, … , 59:12.30},
     1: { 0:12.30, 1:12.30, … , 59:12.30},
     2: { 0:12.30, 1:12.30, … , 59:12.20},
          ---

    59: { 0:12.10, 1:12.00, … , 59:12.00}
  }
}
```

Slika 8. Višestruka očitanja smještena u okviru jednog dokumenta (rezolucija 1 sekunda)

U svakom sistemu može postojati kompromis u pogledu efikasnosti različitih operacija, kao što su ažuriranje (*update*) i unos (*insert*). Tako u nekim sistemima ažuriranje se sprovodi kao kopija originalnih zapisa (*records*) pohranjenih na novu lokaciju i zahtijeva se i ažuriranje indeksa. Jedna od ključnih mogućnosti MongoDB je mehanizam ažuriranja na licu mjesta. Mehanizam ažuriranja, ažuriranjem na nivou polja (*field-level*) se upravlja na mjestu sve dok veličina dokumenta značajno ne naraste. Izbjegavajući prepisivanje cijelog dokumenta i nepotrebni unos indeks stavki, daleko manje se izvodi disk I/O operacija. Obzirom da je ažuriranje na nivou polja efikasnije može se dizajnirati aplikacija u kontekstu ove prednost, tako da dokument orijentisan dizajn ima daleko više ažuriranja (1 za minutu) nego inserta (1 za sat). Pri tome ako želimo da koristimo brojač u aplikaciji, inkrementiranjem a zatim i upisom vrijednosti nazad u bazu podataka može se jednostavno povećati vrijednost polja (npr. 0÷59) koristeći operator $inc. Ovaj pristup ima određen broj prednosti - inkrement operacija je atomska – višestruki tredovi konkurentno mogu izvršiti inkrementaciju korištenjem $inc. Pored toga ovaj pristup je efikasniji u kontekstu broja operacija sa diskom, zahtijeva da manje podataka bude poslano kroz mrežu i zahtijeva manje „kružnih“ putovanja zbog izostavljene potrebe za čitanjem. To su tri značajne pobjede koje rezultiraju u jednostavnijem, efikasnijem i skalabilnijem sistemu.

V. ZAKLJUČAK

U radu je prikazana kontekstualna osnova NoSQL dokument orijentisanih baza podataka i način smještanja podataka sa osvrtom na specifičan tip podataka, podatke vremenskih serija sa senzora kao izvora podataka, a sve ilustrovano kroz prizmu mogućnosti MongoDB - jedne od NoSQL dokument baza podataka.

Ono što je izvjesno i nesporno je da su primjeri realnosti različiti i kompleksni, svaki projekat po sebi u biti je drugačiji, nov izazov, te stoga ne postoji način na koji se može napisati jednostavan algoritam odlučivanja za finalan izbor „idealnih“ skladišta podataka, posebno u vremenu koga karakterišu strukturna raznovrsnost i velike količine podataka. U tom kontekstu ovaj rad daje prikaz aktuelne alternative skladištenja podataka sa namjerom da se ona podrobnije predstavi u svjetlu jedne od reprezentativnih i ozbiljnih opcija skladištenja podataka.

LITERATURA

[1] L. Adam, J. Mattson, "Investigating storage solutions for large data: A comparison of well performing and scalable data storage solutions for real time extraction and batch insertion of data" CHALMERS UNIVERSITY OF TECHNOLOGY Goteborg, Sweden, 2010.

[2] O. Jankovic, “NoSQL model podataka: Kako modelirati grafovski orijentisano”, XXI naučna i biznis konferencija YU INFO 2015, prihvaćen za objavljivanje.

[3] K. Grolinger, W.A. Higashino, A. Tiwari, M.A.M. Capretz , "Data management in cloud environments: NoSQL and NewSQL data stores". JoCCASA, Springer, 2013.

[4] The MongoDB Manual, 2014. MongoDB,Inc. http://docs.mongodb.org/manual

[5] S. Parikh, K. Stirman, “Schema Design for Time Series Data in MongoDB”, 2013., http://blog.mongodb.org/post/65517193370/schema-design-for-time-series-data-in-mongodb

ABSTRACT

When selecting and designing appropriate data model, it is necessary, among other things to take into account the characteristic of the data structure. The sensor data are in that sense really specific, it looks for models in which be presented the nature of the data in the best possible way, especially in the context of the needs of their integration with other data. In this paper will be presented the concept of document-oriented data storage, and one way of storing sensor data based on documents, using NoSQL document database MongoDB.

NoSQL DOCUMENT DATABASE: DATA STORAGE PRESENTATION WITH REFERENCE TO DATA FROM SENSOR

Olivera Janković