Agregatni modeli podataka

Dejan Radić
Codaxy d.o.o.
Banja Luka, RS-BiH
dejanradic@live.com

Sažetak— **Problem skladištenja i obrade velike količine podataka je opisan terminom Big Data. Podskup te klase problema je skladištenje podataka koje se u poslednje vrijeme efikasno rješava primjenom NoSQL baza podataka. Te baze podataka su realizovane sa fokusom na horizontalnoj skalabilnosti, odnosno distribuciji podataka. Agregatni modeli podataka se, zbog svojih karakteristika, lako mogu uklopiti u distribuirano okruženje. Mnoge NoSQL baze podataka implementiraju ove modele podataka. Objašnjen je pojam agregata kao i mogućnost vršenja denormalizacije u procesu modelovanja. Prezentovane su osnovne vrste agregatnih modela podataka sa primjerima. Takođe je vršeno poređenje sa, danas dominantnim, relacionim modelom podataka. Opisana je problematika modelovanja agregatnih modela podataka. Takođe su analizirane mogućnosti modelovanja agregatnih modela podataka preko postojećih notacija za modelovanje (UML, IE, IDEF1X).**

Ključne riječi- Big Data, NoSQL, relacione baze podataka, agregat, denormalizacija, modelovanje podataka

I. UVOD

Kolekcija koncepata koja se koristi za opis podataka u bazi, relacija između podataka, značenja podataka i ograničenja nad podacima se naziva model podataka (eng. *data model*) ili metamodel [1]. Sa korisničkog aspekta, model podataka predstavlja interfejs preko koga se vrši manipulacija podacima. Projektovanje baza podataka se vrši preko apstrakcija koje model podataka nudi. Model podataka se razlikuje od načina skladištenja (eng. *storage model*) ili fizičkog modela podataka, koji podrazumijeva način interne organizacije podataka ili strukturu podataka. Model podataka se može posmatrati kao logička, a način skladištenja kao fizička struktura. Najpoznatiji model podataka je relacioni model. On se sastoji od tabela i njihovih veza, a svaka tabela ima jednu ili više kolona. Jedinica podataka je red u tabeli, koji predstavlja uređenu n-torku (eng. *tuple*).

NoSQL (eng. *Not Only Structured Query Language*) baze podataka implementiraju modele podataka, koji se, po različitim karakteristikama, razlikuju od relacionog modela podataka. Fokus ovog rada je na agregatnim modelima podataka (eng. *aggregate data models*), izostavljajući grafovski model podataka, koji se ne uklapa u agregatnu koncepciju. Većina NoSQL baza podataka rješava problem efikasne distribucije podataka, koja koristi mogućnosti agregatnih modela da se izvrši podjela podataka.

U drugom poglavlju je definisan pojam agregata i opisana mogućnost vršenja denormalizacije. Takođe su poređene karakteristike agregatnih modela podataka sa relacionim. Poslije toga su prikazane karakteristike različitih agregatnih modela podataka, što uključuje ključ/vrijednost, dokument i kolonski model podataka. U šestom poglavlju su analizirane mogućnosti modelovanja agregatnih modela podataka primjenom postojećih notacija za modelovanje podataka. Na kraju je dat zaključak i spisak korištene literature.

II. DEFINICIJA I POREĐENJE SA RELACIONIM MODELOM PODATAKA

Torka je jednostavna struktura koja je definisana skupom vrijednosti bez mogućnosti kreiranja ugniježđenih torki. Odnosno, nije moguće kreirati hijerarhijske strukture. Često se javlja potreba za kreiranjem kompleksnih struktura, gdje lista vrijednosti pripada nekoj drugoj vrijednosti.

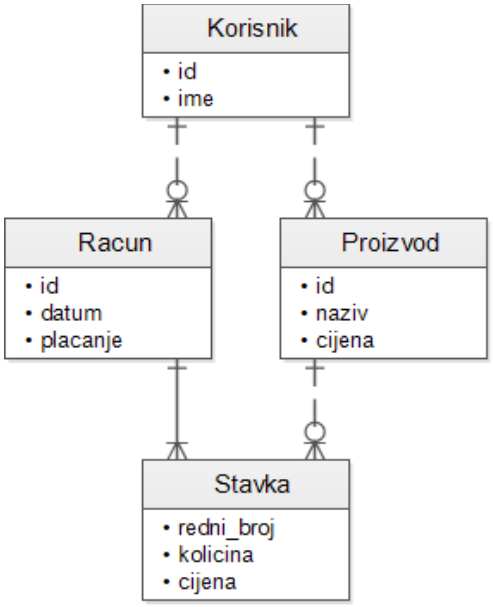
Domenom vođeno projektovanje ili DDD (eng. *Domain Driven Design*) ima definisan pojam agregata. Agregat predstavlja kolekciju objekata ili članova koje ograničava ili enkapsulira korijen agregata. Korijen je jedini član agregata koji je eksterno vidljiv i može se referencirati. Korijen garantuje konzistentnost izmjena u agregatu, tako što eksternim objektima zabranjuje referenciranje članova agregata. Ostali članovi (osim korijena) mogu imati reference jedni na druge [2]. Unutrašnja struktura agregata se može sastojati od polja, odnosno logički nezavisnih vrijednosti.

Drugim rječima, agregat je jedinica za manipulaciju podacima ili za ulazno-izlazne operacije. Te operacije su u opštem slučaju atomične. Posljedica toga je da se garantuje konzistentnost podataka u okviru agregata, kao nedjeljive cjeline. Drugi aspekt je mogućnost distribuiranja agregatno-orijentisanih podataka. Skup agregata se može podijeliti po određenom kriterijumu, tako da se dobijeni podskupovi smještaju na različite čvorove u distribuiranom računarskom sistemu. Svi agregatni modeli podataka podrazumijevaju ključ koji identifikuje agregat, odnosno igra ulogu primarnog ključa. U većini slučajeva, ključ agregata određuje na koji čvor će agregat biti upisan.

Agregatni modeli podataka su: model podataka tipa ključ/vrijednost, model podataka tipa dokument i kolonski model podataka. Na bazi navedenog, razlikuju se tipovi organizacije NoSQL baza podataka sa agregatnim modelima podataka: ključ/vrijednost baze podataka, dokument-orijentisane baze podataka i kolonski orijentisane baze podataka. Agregatni modeli podataka su modeli podataka bez šeme (eng. *schemaless*), što podrazumijeva da različiti, logički slični, agregati mogu imati različitu internu strukturu.

Agregatni modeli podataka su dobili na važnosti zbog povećanja popularnosti NoSQL baza podataka.

Kompleksni agregati kod relacionih baza podataka se formiraju tako što se podaci uzimaju iz više različitih tabela, operacijom spajanja (eng. *join*) tabela. Kada podatke treba sačuvati, radi se dekompozicija kompleksnog dokumenta, tako da se odgovarajući podaci sačuvaju u za to predviđene tabele. Ako je relaciona šema normalizovana (npr. nalazi se u BCNF - *Boyce–Codd Normal Form*), onda pomenute operacije dekompozicije i formiranja agregata mogu biti kompleksne sa aspekta implementacije i izvršavanja (performansi).



Slika 1. Šema evidencije računa

Na Sl. 1 je prikazana šema relacione baze podataka koja reprezentuje činjenice o računima, stavkama na računima i proizvodima. Korisnik može biti prodavac, koji može da kreira ili izda račun sa više stavki, ili administrator sistema, koji može da kreira zapis o novom proizvodu. Račun ima više stavki koje su vezane za proizvod. Račun karakterišu datum kreiranja i način plaćanja (gotovina, kartica itd.), dok proizvod karakterišu naziv i cijena. Stavku karakterišu redni broj, količina i cijena.

Agregati se mogu formirati označavanjem određenih veza kao agregatnih. Veze koje nisu agregatne se nazivaju veze između agregata. Kod većine NoSQL baza podataka sa agregatnim modelima podataka ne postoji sličan mehanizam kao referencijalni integritet koji će spriječiti referenciranje nepostojećeg agregata ili zapisa. Prilikom modelovanja agregata, uzima se u obzir način pristupanja podacima. Potrebno je odrediti kojim entitetima će se pristupati u cjelini, i na bazi toga odrediti agregate. Da bi se odredili agregati, podatke treba posmatrati sa različitih aspekata i odabrati onaj aspekt sa koga će se podaci najčešće posmatrati. To znači da se u toku modelovanja treba spustiti na nivo implementacije i postaviti pitanje: "Koji upiti će se najčešće izvršavati?". U toku modelovanja relacionih baza podataka se posmatraju veze između entiteta, dok se pri modelovanju agregata traži najpogodniji aspekt pristupa podacima. Na prikazanoj šemi je vidljivo da se stavkama neće pojedinačno pristupati, tj. da je stavka dio računa ili slabi entitetski tip (postoji egzistencijalna zavisnost ili identifikujuća veza). To znači da je veza između računa i stavke agregatna. Veza između proizvoda i stavke (računa) se označava kao veza između agregata. Činjenica da se više stavki nalazi u okviru jednog računa se može reprezentovati pomoću JSON (eng. *JavaScript Object Notation*) formata, na sljedeći način:

```
{
    id: 5,
    korisnik_id: 1,
    datum: "01-01-2016",
    placanje: "gotovina",
    stavke: [{
        id: 30,
        proizvod_id: 5,
        komada: 3
    },{
        id: 31,
        proizvod_id: 12,
        komada: 1
    }]
}
```

Entitet koji se logički nalazi na vrhu hijerarhije agregata se naziva korijen agregata. Ostali pripadajući entiteti su logički objedinjeni u korijenu agregata. Ako bi se veza između korisnika i računa označila kao agregatna, onda bi se stvorila tri nivoa (korisnik, račun i stavka). U tom slučaju je korisnik korijen agregata. Ako bi se sve veze označile kao agregatne, onda bi se podaci o stavkama duplirali, jer bi pored računa, koji ima pripadajuće stavke, i proizvod imao sve stavke koje su sa njim asocirane. Takođe, slabi entitetski tip kao što je stavka može da referencira (vezom između agregata) neki drugi jaki entitetski tip, kao što je proizvod. Obrnuto nije u skladu sa definicijom agregata. Zato, suštinski gledano, nije moguće definisati agregatnu vezu između korisnika i računa, a da se ne definiše agregatna veza između računa i stavke.

Vidljivo je da se dobavljanjem agregata iz primjera neće dobiti svi potrebni podaci da bi se predstavio jedan račun. Račun može imati specifikovano ime korisnika (prodavca) koji ga je kreirao ili izdao, a stavke trebaju da maju naziv proizvoda. Dobavljanje tih podataka zahtijeva traženje drugih agregata preko veza između agregata. Za slučaj većeg broja stavki bi se tražio veći broj agregata koji definišu proizvod. U pitanju je operacija slična operaciji spajanja kod relacionih baza podataka. S obzirom da su agregatni modeli podataka dizajnirani sa fokusom na distribuciju podataka, operacija spajanja je, zbog složenosti izvršavanja u distribuiranom okruženju, uglavnom izostavljena kod NoSQL baza podataka koje implementiraju agregatne modele podataka. Pristup rješavanju ovog problema postoji i kod relacionih baza podataka, a to je denormalizacija.

Normalizacija se postiže dovođenjem šeme relacione baze podataka u odgovarajuću normalnu formu. Posljedica je da neće biti redundancije ili duplikacije podataka, kao ni anomalija pri ažuriranju ili brisanju zapisa. S druge strane, da bi se postigla data organizacija, mnoge relacije (tabele) se dekomponuju, što rezultuje većim brojem relacija. Eliminacija redundancije dekompozicijom može ponekad "previše" da fragmentira organizaciju baze podataka, što može da ima za posljedicu povećanu kompleksnost specifikacije upita,

višestruka spajanja relacija i degradaciju performansi [1]. Generalno, što više normalizovanih relacija postoji u bazi podataka, potrebno je više spajanja relacija da bi se rekonstruisali kompletni logički entiteti [3]. Na bazi navedenog, u određenim slučajevima se javlja potreba za denormalizacijom relacione šeme. To dovodi do povećane količine podataka, ali i do eliminacije potrebe za spajanjem, ako je denormalizacija obavljena na odgovarajući način. Na taj način se povećavaju peformanse upita. U navedenom primjeru bi se u račun dodala kolona za ime korisnika, koji je kreirao dati račun. Ako se ime korisnika izmjeni u tabeli korisnik, onda se to treba propagirati i do računa koje je kreirao taj korisnik, ako je to potrebno. To se postiže najčešće transakcionim ponašanjem na nivou aplikacije ili trigerima na nivou baze podataka. Opisani način denormalizacije je poznat pod nazivom pre-join.

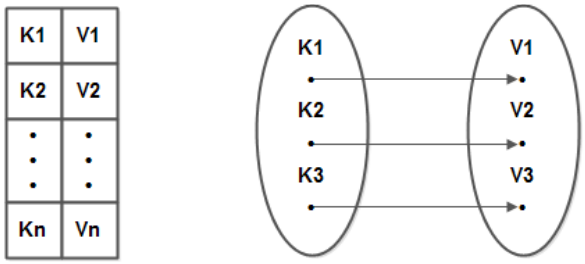
Ako se denormalizacija ne bi vršila kod agregatnih modela podataka, to bi podrazumijevalo uzimanje podataka iz drugih agregata, odnosno prelaženje preko veza između agregata da bi se dobili dati podaci. U opštem slučaju se pri kreiranju datog agregata dobave svi potrebni podaci da bi se dobio potpun zapis. U navedenom primjeru stavka dobija naziv proizvoda i njegovu cijenu. Izmjena naziva proizvoda može povući izmjenu denormalizovanog naziva u svim stavkama. Izmjena cijene proizvoda ne bi trebalo da utiče na cijene stavki, jer su u pitanju istorijski podaci. S obzirom da se stavka po definiciji agregata ne može referencirati, identifikator je izostavljen. Stavke se u okviru proizvoda nalaze u formi liste, gdje je definisan poredak, tako da nije potrebno polje koje će definisati redni broj. Ukupan iznos na računu se može izračunati iz podataka koji postoje u stavkama. Da se ukupan iznos ne bi računao pri svakom dobavljanju računa, on se može izračunati pri inicijalnom kreiranju računa i čuvati kao polje u računu. U tom slučaju se radi o optimizaciji, a ne denormalizaciji. Opisane izmjene primjenom denormalizacija i optimizacije su predstavljene JSON formatom:

```
{
    id: 5,
    korisnik_id: 1,
    korisnik_ime: "Administrator",
    datum: "01-01-2016",
    placanje: "gotovina",
    ukupno: 15.5,
    stavke: [{
        proizvod_id: 5,
        proizvod_naziv: "AAA Baterije",
        komada: 3,
        cijena: 4.5
    },{
        proizvod_id: 12,
        proizvod_naziv: "Sveska",
        komada: 1,
        cijena: 2.0
    }]
}
```

III. MODEL PODATAKA TIPA KLJUČ/VRIJEDNOST

Kompleksni upiti, analiza podataka i njihovih veza iz različitih aspekata uglavnom pripadaju domenu relacionih baza podataka. Za aplikacije je ponekad bitno da se podaci dobiju jednostavno i brzo, bez potrebe da se podaci transformišu. Model podataka kod ključ/vrijednost baza podataka je jednostavan. Vrijednosti ili agregatu se pristupa preko nepromjenljivog (eng. *immutable*) ključa. To je ekvivalentno pristupu preko primarnog ključa kod relacionih baza podataka. Sa matematičkog aspekta, u pitanju je binarna relacija između skupa ključeva i skupa vrijednosti. Ključ predstavlja jedinstven podatak, koji je inače tipa string. Vrijednost je BLOB (eng. *Binary Large Object*), odnosno niz bita koji ima semantičku vrijednost za korisnika. Ključ/vrijednost baze podataka, kao što su Riak i Redis, implementiraju ovaj model podataka.

Model podataka tipa ključ/vrijednost je agregatni model podataka. Baze podataka koje implementiraju ovaj model podataka se nazivaju ključ/vrijednost baze podataka. U ovom slučaju agregat je vrijednost. Jedna od važnih karakteristika je potpuna sloboda po pitanju sadržaja agregata ili vrijednosti. To znači da se ne zalazi u strukturu vrijednosti. Jedino ograničenje koje može da postoji jeste memorijski prostor za smještanje vrijednosti. Slična struktura podataka je dostupna u programskim jezicima pod nazivom mapa, rječnik ili asocijativni niz. Na sličan način se realizuje i komunikacioni interfejs sa bazama podataka koje implementiraju ovaj model podataka. Kolekcije uređenih parova (ključ, vrijednost) se grupišu u imenske prostore (eng. *namespace* ili *bucket*), koji daju mogućnost separacije logički različitih podataka.



Slika 2. Model podataka tipa ključ/vrijednost

IV. MODEL PODATAKA TIPA DOKUMENT

Model podataka tipa ključ/vrijednost ima jednostavan komunikacioni interfejs. Vrijednosti se pristupa pomoću ključa, bez zalaženja u strukturu ili sadržaj vrijednosti. Ponekad je potrebno da se filtriraju vrijednosti po nekom kriterijumu. Model podataka tipa dokument podrazumijeva dokumente čijoj se strukturi može pristupiti. Dokument se sastoji od polja i drugih dokumenata. Ovaj model podataka je agregatni model podataka (agregat je dokument). Dokumentu se, kao i kod modela podataka tipa ključ/vrijednost, pristupa preko ključa. Razlika u odnosu na model podataka tipa ključ/vrijednost je mogućnost filtriranja dokumenata ili agregata po poljima (restrikcija). To znači da model podataka tipa dokument ima transparentne podatke u agregatu. Dokument-orijentisane baze podataka, kao što je MonogDB, implementiraju ovaj model podataka.

U pitanju je fleksibilan model podataka, kojim se mogu predstaviti mnoge realne situacije. Potpuno se uklapa u hijerarhijsku strukturu koja je osnova agregatnih modela podataka. Dokumenti se grupišu u kolekcije. Opisana hijerarhijska struktura se može predstaviti različitim formatima: XML, JSON, YAML, BSON itd. U tabeli I je dat

primjer dokumenta za već opisani primjer računa i stavki na računu. Stavke predstavljaju ugniježđeni niz u okviru računa. Svaka stavka je zaseban dokument koji može imati različita polja.

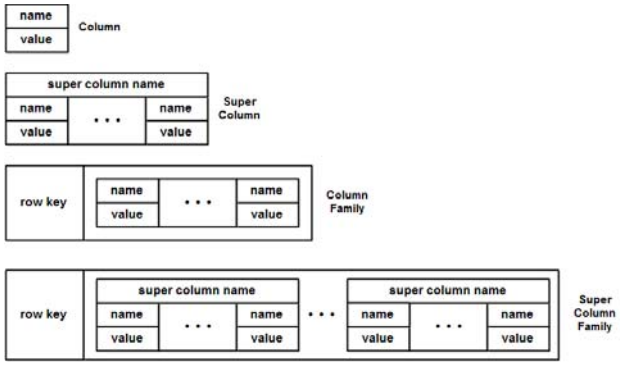
TABELA I. PRIMJER DOKUMENTA

Ključ	Dokument
5	korisnik_id: 1 korisnik_ime: "Administrator" datum: "01-01-2016" placanje: "gotovina" ukupno: 15.5 stavke: proizvod_id: 5 proizvod_naziv: "AAA Baterije" komada: 3 cijena: 4.5 proizvod_id: 12 proizvod_naziv: "Sveska" komada: 1 cijena: 2.0

V. KOLONSKI MODEL PODATAKA

Kolonski model podataka organizuje podatke u redove i kolone. Način organizacije podataka je prvobitno inspirisan Google BigTable konceptom. Prava moć kolonskog modela je u denormalizovanom pristupu strukturisanju rijetkih podataka [3]. U pitanju je agregatni model podataka koji ima mnoge sličnosti sa relacionim modelom podataka. Razlika je što redovi ne moraju imati iste kolone, odnosno, radi se o modelu podataka bez šeme. Ako postoji veliki broj kolona, a redovi imaju samo one koje su im potrebne, onda je u pitanju rijetka (eng. *sparse*) tabela.

Kolona (eng. *column*) je uređeni par (naziv, vrijednost), gdje se naziv može posmatrati kao ključ ili identifikator kolone. Red (eng. *row*) je kolekcija kolona koje su povezane određenim ključem koji identifikuje red (eng. *row key*), što predstavlja sličnost sa modelom podataka tipa ključ/vrijednost. Više sličnih redova čini kolonsku familiju (eng. *column family*). Ako jedna kolona može imati jedan nivo ugniježđenih kolona, onda je u pitanju super kolona (eng. *super column*). Korištenjem super kolona za kreiranje kolonskih familija se dobija super kolonska familija (eng. *super column family*), kao što je prikazano na Sl. 3. Kolonske familije i super kolonske familije se smještaju u prostore ključeva (eng. *keyspaces*). Model podataka tipa dokument se razlikuje od modela podataka tipa ključ/vrijednost po mogućnosti vršenja restrikcije. Kolonski model podataka se razlikuje od modela podataka tipa dokument po mogućnosti vršenja projekcije na efikasan način. Kolonske baze podataka, kao što je Cassandra, implementiraju ovaj model podataka.



Slika 3. Kolonski model podataka

VI. MODELOVANJE AGREGATNIH MODELA PODATAKA

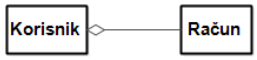
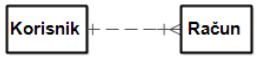
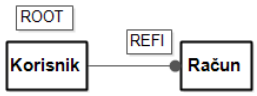
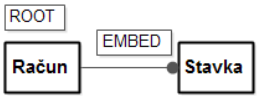
Trenutno ne postoje standardi za modelovanje NoSQL baza podataka [4]. Proizvođači NoSQL SUBP u okviru dokumentacije najčešće imaju smjernice za modelovanje u okviru njihovog rješenja. Za modelovanje entiteta se koriste različite notacije, koje propisuju svoje vrste veza između datih entiteta. Agregat može da obuhvata više entiteta građenjem agregatnih veza između njih. Tu su i veze između agregata koje definišu povezanost agreagata ili entiteta u agregatu sa nekim drugim agregatom. Navedene veze se mogu modelovati primjenom različitih notacija za modelovanje. U tabeli II su prikazane UML (eng. *Unified Modeling Language*), IE (eng. *Information Engineering*) i IDEF1X (eng. *Integration DEFinition for Information Modeling*) notacije na primjeru računa i stavke.

UML definiše pomove agreagacije (eng. *aggregation*) i kompozicije (eng. *composition*) koji se mogu uklopiti u agreagatni koncept. UML agregaciju treba razlikovati od pojma agregata po definiciji domenom vođenog projektovanja. UML agregacija predstavlja odnos cjelina-dio (eng. *part-of*) [5]. UML agregacija omogućava modelovanje veze tip cjelina-dio u kojoj dijelovi mogu da egzistiraju bez cjeline, dok u UML kompoziciji dijelovi ne mogu da egzistiraju bez cjeline [1]. Egzistencijalna zavisnost se korištenjem UML-a modeluje kompozicijom. Postoji logička sličnost između agregatne veze i UML kompozicije, i veze između agregata i UML agregacije.

IE notacija se često koristi za konceptualno modelovanje baza podataka. Neidentifikujuća veza se predstavlja isprekidanim linijama. S druge strane, identifikujuća veza se predstavlja punom linijom. Kardinalnost se predstavlja notacijom "ptičije stopalo" (eng. *crow's foot notation*). U primjeru je predstavljena veza 1:1..*, jer svaka stavka treba biti u vezi sa računom, a račun treba imati bar jednu stavku. Identifikujuća veza podrazumijeva da je entitetski tip na strani više slabi entitetski tip. Identifikujuća veza ima sličnost sa agregatnom vezom, dok se veza između agregata može predstaviti neidentifikujućom vezom.

U [6] su predstavljena proširenja za modelovanje agregata primjenom IDEF1X standarda. Prijedlog je da se koristi tradicionalno modelovanje podataka da se razvije konceptualni model, a poslije da se primjene novi operatori koji će taj model prilagoditi agregatnoj formi. Predložena su 3 operatora: ROOT, REFI i EMBED. ROOT operator služi za definisanje korijena agregata.

TABELA II. NOTACIJE ZA MODELOVANJE PODATAKA

	Veza između agregata	Agregatna veza
UML	 Agregacija	 Kompozicija
IE	 Neidentifikujuća veza	 Identifikujuća veza
IDEF1X	 Referenciranje	 Ugniježdenost

REFI operator podrazumijeva referenciranje određenog entiteta. EMBED operator podrazumijeva ugniježđenost entiteta. Vidljiva je sličnost REFI operatora sa vezom između agregata, a EMBED operatora sa agregatnom vezom. REFI se koristi da redukuje graf modela u stablo, a EMBED je potreban da bi redukovao model eksplicitnim ugniježđenjem entiteta [6]. Opisane notacije, zajedno sa njihovim proširenjima, se mogu koristiti za modelovanje agregatnih modela podataka. Međutim, treba uzeti u obzir njihove specifičnosti.

VII. ZAKLJUČAK

NoSQL baze podataka koriste drugačije modele podataka od relacionih baza. U pitanju su agregatni modeli podataka koji su bazirani na definiciji agregata, kao pojma nastalog u okviru domenom vođenog projektovanja. Agregatni modeli podataka su popularizovani rastom potražnje za NoSQL bazama podataka. Jedan od osnovnih razloga nastanka NoSQL paradigme je upravljanje velikom količinom podataka, što je, u većini slučajeva, dovelo do primjene koncepta distribucije podataka. Agregatna orijentacija, zbog svojih karakteristika, omogućava efikasnu distribuciju. Međutim, to dovodi do problema pri vršenju upita koji podrazumijevaju kombinovanje podataka sa različitih čvorova. S druge strane, relacione baze podataka podrazumijevaju spajanje više tabela kada je potrebno formirati kompleksnu strukturu. To može biti vremenski zahtjevna operacija, naročito u slučaju spajanja većeg broja tabela, pa se koristi koncept denormalizacije da bi se parcijalno, ili u potpunosti, izbjegla potreba za spajanjem. Sličan koncept se primjenjuje kod agregatnih modela podataka, da bi se izbjeglo praćenje veze između agregata. Praćenjem tih veza se vrši dodatno dobavljanje podataka, koje može podrazumijevati čitanje podataka sa različitih čvorova u klasteru. Iz tog razloga, NoSQL baze podataka imaju drugačiji način modelovanja, koji uzima u obzir način pristupa podacima. Ne postoji definisan standard za modelovanje NoSQL baza podataka. Jedan od razloga je veliki broj različitih baza podataka, ove vrste, koje se svakodnevno pojavljuju. Međutim, za modelovanje se mogu primjeniti postojeće notacije. Neke već imaju mehanizme da predstave agregatne veze i veze između agregata, dok je druge potrebno proširiti. Različiti agregatni modeli podataka imaju različite mogućnosti u pogledu vršenja upita. Zajednička osobina je pristup sadržaju agregata preko ključa. Model podataka tipa ključ/vrijednost podrazumijeva čitanje podataka preko ključa, bez mogućnosti vršenja projekcije i restrikcije nad sadržajem. Model podataka tipa dokument ima mogućnost vršenja restrikcije nad sadržajem, što podrazumijeva zalaženje u strukturu agregata. Kolonski model podataka može vršiti efikasnu projekciju.

LITERATURA

[1] S. Marić, D. Brđanin, Relacione baze podataka. Elektrotehnički fakultet Univerziteta u Banjoj Luci, BiH, 2012

[2] E. Evans, Domain-Driven Design. Addison-Wesley, USA, 2003

[3] D. McMurtry, A. Oakley, J. Sharp, M. Subramanian, H. Zhang, Data Access for Highly-Scalable Solutions: Using SQL, NoSQL, and Polyglot Persistence. Microsoft, USA, 2013

[4] S. Benson, "Polymorphic Data Modeling," M.S. thesis, C.S., Georgia Southern University, Statesboro, GA, USA, 2014

[5] M. Fowler, UML ukratko. Mikro knjiga, SRB, 2004

[6] V. Jovanovic, S. Benson, "Aggregate Data Modeling Style", Proceedings of the Southern Association for Information Systems Conference, USA, 2013

ABSTRACT

Big Data is that describes problem of storing and processing large data sets. Subset of that problem is data storage, which is efficiently solved by using NoSQL databases. That kind of databases are implemented with focus on horizontal scalability and data distribution. Aggregate data models can be easily used in distributed environment, because of its characteristics. Many NoSQL databases are implementing these data models. Paper contains aggregate definition as well as description of denormalization possibility in modeling process. Basic types of aggregate data models are presented with examples. Aggregate data models were compared with relational data models which are widely used nowadays. Issues about aggregate data modeling were described. Possibilities of aggregate data modeling by using existing data modeling notations (UML, IE, IDEF1X) were analyzed.

AGGREGATE DATA MODELS

Dejan Radić