NoSQL i skladišta podataka

Katerina Černjeka
Sveučilište u Rijeci/ Odjel za informatiku, Rijeka, Hrvatska
kcernjeka@inf.uniri.hr

Sažetak - Trenutni trend NoSQL baza podataka primarno je motiviran aplikacijama iz domene Web 2.0 alata te općenito pojavom velike količine različitih tipova podataka. Neki zahtjevi za pohranu podataka aplikacija nadilaze mogućnosti relacijskih baza podataka okarakteriziranih ACID svojstvima i formalnom relacijskom shemom. U kontekstu skladišta podataka pojava velike količine različitih tipova podataka, osim problema pohrane, manifestira se i povećanjem zahtjeva za analizom podataka. Međutim, ta analiza podataka više ne uključuje samo korporativne podatke transakcijskih baza podataka, već sve podatke iz kojih se mogu izvući korporativne vrijednosti, uključujući tekst i nestrukturirane podatke. U ovom radu je napravljena usporedba relacijskih i NoSQL baza podataka, navedene su osnove karakteristike te su opisani modeli pohrane podataka i upitni jezik NoSQL baza podataka. Predstavljeni su osnovni koncepti skladišta podataka te pregled istraživanja u kontekstu skladišta podataka i NoSQL baza podataka. Definirane su smjernice za daljnje istraživanje u području NoSQL baza podataka i skladišta podataka uz naglasak na integraciju podataka iz NoSQL izvora.

Ključne riječi: ***NoSQL, model podataka, baze podataka, skladišta podataka*****.**

I. UVOD

Još od 80 – tih godina relacijski model podataka i Codd-ovi temelji relacijske baze podataka dominiraju u području pohrane i upravljanja podacima preko jedinstvenog SQL jezika. Nastale su u vremenu velikih računala i poslovnih aplikacija – puno prije popularizacije Interneta, računarstva u oblaku (eng. *cloud*), podataka velikog opsega (eng. BigData), mobilnih usluga i digitalne ekonomije. Veliki proizvođači poput Oracle-a, IBM-a i Microsoft-a nude niz sustava za upravljanje relacijskim bazama podataka.

Carlo Strozzi [1] 1998. godine prvi put spominje ime NoSQL kao naziv relacijske baze otvorenog koda, bez SQL sučelja. Ime je dobila zbog činjenice da ne koristi SQL jezik za postavljanje upita nad podacima. Nakon toga tek 2009. godine Eric Evans [2] predstavlja naziv NoSQL na konferenciji baza podataka otvorenog koda održanoj u San Franciscu.

Pojavom velike količine podataka generiranih od strane mnogobrojnih korisnika dovelo je do sve više slučajeva u kojima relacijske baze ne odgovaraju potrebama sustava i korisnika. Najviše problema uzrokuje model podataka definiran formalnom relacijskom shemom. Relacijska baza podataka dizajnirana je pretežito za rad na jednom serveru – čim većem i jačem, tim boljem. Time se ona može samo skalirati prema gore (eng. *scale up*) dodavanjem više procesora, diskovnog prostora i memorije, ali ne i skalirati prema van (eng. *scale out*) korištenjem više slabijih računala spojenih u računalnu mrežu – klaster (eng. *cluster*). Klaster slabijih računala koristi hardver generalne primjene kao jeftiniji pothvat za eksponencijalni rast količine podataka [3]. Prednost korištenja takvog sustava klastera je rješavanje problema otpornosti na greške. Sustav se može podesiti da nastavi sa radom i ukoliko je neko individualno računalu u kvaru, što osigurava visoku dostupnost podataka.

Formalna relacijska shema može predstavljati problem i nije prikladna za upotrebu nekim web aplikacijama kao npr. blogovi, zbog pohrane nestrukturiranih podataka (dok relacijske baze podataka pohranjuju samo strukturirane podatke). U blogu je sadržan veliki broj različitih vrsta podataka kao što su tekst, komentari, slike, video, izvorni kod i ostale relevantne informacije koje se pohranjuju kroz različite tablice. Takve aplikacije su relativno fleksibilne pa se i od pozadinske baze podataka isto tako očekuje fleksibilnost i mogućnost prilagodbe sheme. Dodavanje ili brisanje značajki bloga nije moguće bez privremene nedostupnosti sustava ako koristimo relacijsku bazu podataka [4]. Kao rješenje navedenog problema neprikladnosti relacijske sheme autori u [5] predstavili su metodologiju za prelazak sa relacijske baze podataka (MySQL) na NoSQL bazu podataka (MongoDB).

Problem povećanja količine podataka i njihove pohrane u digitalnom svijetu prepoznale su i velike organizacije kao Facebook, Amazon i Google. Era NoSQL baza podataka započela je objavom nekoliko značajnih radova koji su uzrokovali da NoSQL baze podataka postanu konkurent relacijskih baza podataka. Glavni predvodnik je Google organizacija koja do 2006. godine izdaje niz dokumenata [6], [7] i [8]. Nakon Google-a popularizaciji netradicionalnog pristupa pohrane i obrade podataka, pridružuje se i Yahoo! organizacija koja razvija sustav distribuirane infrastrukture otvorenog koda naziva Hadoop. 2007. godine Amazon objavljuje Dynamo [9], svoj nerelacijski, visoko skalabilni distribuirani sustav za pohranu podataka.

Google i Amazon su nastojali riješiti problem količine podataka brojive u petabajtima te masivne zahtjeve čitanja i pisanja u bazu koji moraju biti izvršeni bez primjetnog kašnjenja. Kako bi se mogle nositi sa takvim zahtjevima, organizacije održavaju klastere sa tisućama računala. Distribuiranom arhitekturom postižu velike performanse, visoku dostupnost podataka te mehanizam upravljanja greškama ili pada sustava kako bi informacije korisnicima bile pravovremeno dostupne.

Normalizirani model podataka i ACID svojstva relacijskih baza podataka imaju negativan utjecaj na performanse i dostupnost sustava u distribuiranom okruženju eksponencijalnog porasta podataka. Pritom mislimo na sljedeća ACID svojstva: atomarnost (eng. *Atomicity*), konzistentnost (eng. *Consistency*), izolacija (eng. *Isolation*) i trajnost (eng. *Durability*). Izostavljanjem jednog ili više ACID svojstava postižu se veće performanse i skalabilnost sustava.

NoSQL baze podataka zapostavljaju komponentu *konzistentnosti* ACID svojstva pa zato kažemo da su bazirana na BASE svojstvima [10]:

- načelna dostupnost (eng. *Basic Availabile*) – sustav je stalno dostupan, svaki zahtjev dobiva odgovor
- labavo stanje (eng. *Soft state*) - stanje sustava je promjenjivo, čak i u slučaju kada nemamo eksplicitne upite nad bazom podataka (zbog naknadne konzistentnosti stanje konzistentnosti podataka se često mijenja).
- naknadna konzistentnost (eng. *Eventually consistent*) – sustav će s vremenom postati konzistentan.

U [11] Eric Brewer predstavlja CAP teorem koji se sastoji od 3 glavna svojstva:

- konzistentnost (eng. *Consistency*), definirana na način da postoji jedinstvena up-to-date kopija podataka
- visoka dostupnost (eng. *Availability*) podataka (za ažuriranje),
- tolerancija particioniranja (eng. *Partition tolerance*) koja omogućava sustavu nesmetan rad iako je pojedini dio sustava nedostupan.

Osnovna ideja teorema je da distribuirani sustav ne može istovremeno zadovoljiti sva 3, već samo 2 navedena svojstva. U praksi NoSQL baze koriste sve značajke CAP teorema, ali sa manje strogim zahtjevima od ovih navedenih u teoremu.

Prema CAP teoremu preliminarna klasifikacija NoSQL baza podataka je sljedeća [12]:

- Baze podataka orijentirane na konzistenciju i visoku dostupnost (*CA*) – dio baze podataka ne uključuje toleranciju particioniranja te koristi pristup repliciranja (eng. *Replication approach*) kako bi osigurali navedena svojstva (pretežito relacijske baze podataka).
- Baze podataka orijentirane na konzistenciju i toleranciju particioniranja (*CP*) – podaci su najčešće pohranjeni u čvorovima, osigurana je konzistencija ali nema dobre podrške za visoku dostupnost podataka.
- Baze podataka orijentirane na visoku dostupnost i toleranciju particioniranja (*AP*)

Autori iz [13], [4], [14] navode nekoliko ključnih svojstava NoSQL baza podataka:

1. Jednostavan i fleksibilan nerelacijski model podataka dizajniran za obradu mnogo različitih struktura podataka
2. Horizontalno skaliranje na više servera
3. Repliciranje i particioniranje podataka na više servera
4. Jednostavni i laki pozivi sučelja i protokola
5. Slabiji model konkurentnosti (eng. *Concurrency model*) od relacijskog sustava za upravljanje bazom podataka
6. Upotreba distribuiranih indeksa i RAM-a za efikasniju pohranu podataka
7. Dinamičko dodavanje novih atributa

II. MODEL POHRANJIVANJA PODATAKA NOSQL BAZA PODATAKA

Ovisno o modelu podataka NoSQL baze podataka možemo podijeliti u 4 osnovne kategorije [15], [4], [14], [16], [17], [18]: ključ-vrijednost baze podataka, dokument baze podataka, stupčane baze podataka i graf baze podataka.

A. Ključ-vrijednost baze podataka

Najjednostavniji od 4 navedena modela podataka. Podržava pohranu podataka bez naglaska na shemu podataka. Informacije pohranjene u sustav tretirane su kao asocijativno polje ulaznih vrijednosti, sastavljeno od jedinstvenog ključa i asociranog podatkovnog djela. Jedinstveni ključ identificira ulaz a podatkovni dio predstavlja vrijednost, zajednički čine par ključ-vrijednost [15].

Ključ-vrijednost model podataka (*Slika 1*) možemo interpretirati kao složeniju modifikaciju pohrane podataka uz pomoć asocijativnog polja. Asocijativno polje je polje sa manje strogim zahtjevima na tip vrijednosti koju pohranjujemo. Vrijednost indeksa polja nisu ograničene na cijeli broj (tip integer), a vrijednosti koje pohranjujemo mogu biti različitih tipova. Bitno je da svaka vrijednost ima jedinstveni identifikator (ključ) unutar jednog imenika. Imenik je skup ključ-vrijednost parova međusobno logički povezanih.

Glavna prednost takvog sustava za upravljanje bazom podataka je primarno njegova jednostavnost (modela i sintakse za upravljanje podacima), mogućnost brzog izvođenja velikog broja upita te skalabilnost.

Ključ-vrijednost baze podataka možemo podijeliti na *in-memory* baze koje podatke pohranjuju u memoriju (Memcached [19] i Redis [20]) ili one baze koje podatke čuvaju dugoročno na trajnim spremištima za pohranu podataka (eng. *persistant key-value stores*) [16] (BerkleyDB [21], Voldemort [22] i Riak [23]).

BANK DATABASE	
Key	Value
1	ID:1 Joining Date: 15-July-1985 Designation: Cashier
2	ID:2 Joining Date: 19-March-1982 Designation: Manager
3	ID:3 Joining Date: 4-April-1988 Designation: Front Desk Officer

Slika 1: Model ključ – vrijednost baze podataka [18]

B. *Stupčane baze podataka*

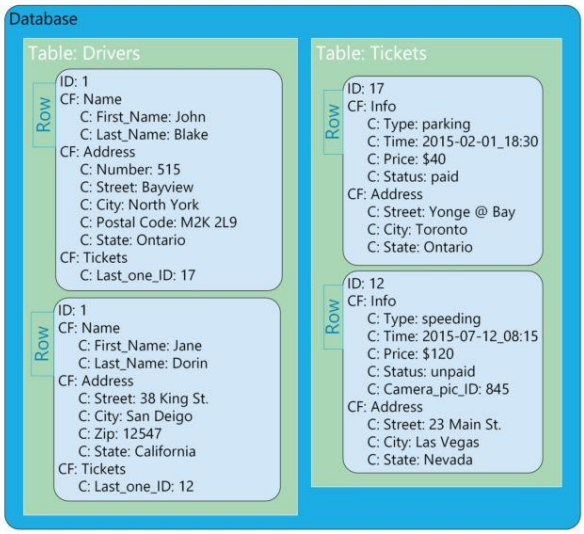
Kreator stupčanih baza podataka je Google i njegov BigTable distribuirani sustav za pohranu i upravljanje strukturiranim podacima namijenjenim skaliranju na visoke razine: petabajti podataka nad tisućama računala [8]. Inženjeri koji su osmislili Google BigTable očekivali su potrebu za upravljanjem desecima tisuća stupaca te milijardom redaka. Zahtjevi uključuju veliku brzinu pristupa podacima, visoku dostupnost preko više podatkovnih centara te fleksibilnost upravljanja podacima. Kao rezultat, stupčani sustavi imaju neke sličnosti s ključ-vrijednost bazama podataka, dokument bazama podataka i relacijskim bazama podataka [3].

Entiteti su modelirani recima (1 redak je 1 entitet) a svaki redak ima svoj jedinstveni identifikator, kao i primarni ključ u relacijskim bazama podataka. Svaki je stupac jedinstveno određen imenom, a svaka vrijednost unutar stupca može sadržavati više različitih vrijednosti označenih drugačijom vremenskom oznakom. Vrijednosti stupaca se ne mijenjaju već se samo dodaju nove vrijednosti s različitim vremenskim oznakama.

Za razliku od relacijskih baza podataka koje spremaju sve vrijednosti pojedinog retka zajedno, stupčane baze podataka spremaju zajedno samo dijelove redaka. Jedan redak može sadržavati više familija stupaca. Familije stupaca su logičke grupacije stupaca. Familije stupaca vezane uz određen redak mogu biti pohranjene zajedno a stupci unutar iste familije stupaca uvijek su pohranjeni zajedno (S*lika 2*).

Osnovni model pohrane podataka koji BigTable koristi organizira podatke kroz 3 dimenzije: retke, stupce i vremenske oznake. URL adrese stranica predstavljaju ključeve redaka, različiti aspekti web stranica predstavljaju nazive stupaca a presjek retka i stupca je sadržaj web stranice pojedinih aspekata (za vremensku oznaku dohvaćanja sadržaja).

Najpoznatije stupčane baze podataka su Hbase [24], Hypertable [25] kao implementacija otvorenog koda sustava BigTable i Cassandra [26].



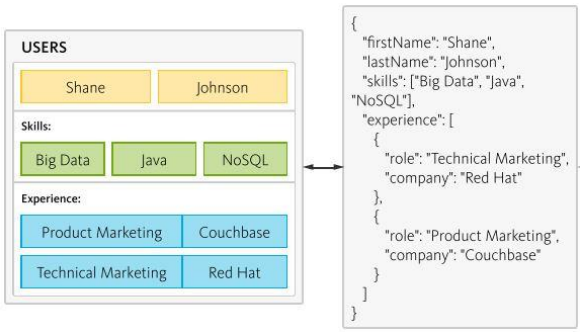
Slika 2: Model stupčane baze podataka [17]

C. *Dokument baze podataka*

Razvojni inženjeri često se okreću ka dokument modelu NoSQL baza podataka kada im je potrebna fleksibilnost ključ-vrijednost modela podataka, ali upravljaju sa složenijim podatkovnim strukturama te im je bitna mogućnost indeksiranja podataka i pretraživanja istih po vrijednosti. Vidjeli smo da ključ-vrijednost baze podataka za svaki ključ vraćaju vrijednost vezanu za taj ključ. Dokument baze podataka rade po drugačijem principu. Ključ dokument baza podatka jest jedinstven identifikator kao u ključ-vrijednost bazama, no on se ne mora koristi za operacije nad bazom. Dohvaćanje objekta iz baze obavlja se postavljanjem upita na sadržaj unutar dokumenta. Dakle dokument baze podataka možemo smatrati složenijom varijantom ključ-vrijednost baza podataka (S*lika 3*), gdje se kao vrijednost spremaju dokumenti koji se još mogu i pretraživati [3].

Za pohranu podataka koristi se JSON (eng. *JavaScript Object Notation*), XML ili BSON format (eng. *Bynary JSON*) [27]. Slične dokumente grupiramo u kolekcije koje čine hijerarhijsku strukturu srodnu stablima.

Najpoznatije dokument baze podataka: CouchDB [28], MongoDB [29] i Riak [23] (ima veze za modeliranje veza između dokumenata).

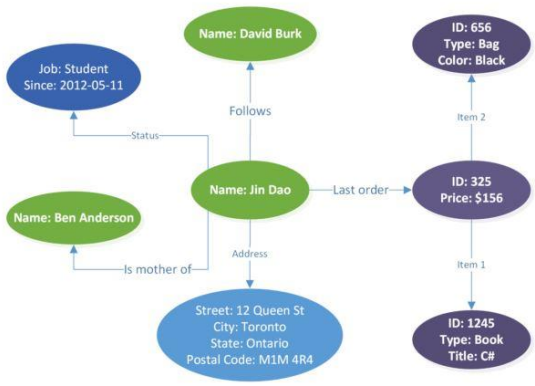


Slika 3: Model dokument baze podataka [30]

D. *Graf baze podataka*

Graf baze podataka posebno su korisne za upravljanje entitetima koji su strogo povezani (*Slika 4*). Grafovi se sastoje od dvije osnovne komponente, čvorova kojima modeliramo entitete te bridova kojima modeliramo veze među entitetima. S obzirom na to koje vrste veza prikazujemo, bridovi mogu biti usmjereni ili neusmjereni. Graf baze podataka slične su objektno - orijentiranim bazama obzirom da su grafovi prikazani kao objektno – orijentirana mreža čvorova (konceptualnih objekata), veza između čvorova (bridovi) i svojstava (atributa objekata prikazanih preko ključ – vrijednost parova). Vizualno predstavljanje i prikaz čvorova i odnosa među njima čine graf baze podataka široko primjenjivima u slučaju kada nas više zanimaju veze među podacima nego sami podaci.

Najpoznatije graf baze podataka: Neo4j [31] i GraphDB [32].



Slika 4: Model graf baze podataka [17]

Autori iz [33] osim 4 osnovne kategorije NoSQL baza navode još i:

E. Objektno – orijentirane baze podataka

Poznatije pod nazivom OODBMS, pohranjuju podatke u obliku objekata te podržavaju nasljeđivanje i višestruko nasljeđivanje, slično kao i u objektno orijentiranom programiranju.

F. Baze podataka na mreži i u oblaku (eng. Grid and Cloud Databases)

Koriste mrežu i oblak za upravljanje heterogenim i geografski distribuiranim bazama podataka te olakšavaju pristup udaljenom hardveru i resursa za pohranu podataka.

G. XML baze podataka

Kao format pohrane koriste XML dokument čiji podaci su često podijeljeni u dijelove koji se zatim pohranjuju u tablice korištenjem XML sloja mapiranja.

H. Višedimenzionalne baze podataka (eng. Multidimensional Databases)

Podatke spremaju kao n – dimenzionalnu matricu. Mogu se koristiti relacijske baze podataka u backendu u kojima se višedimenzionalni upiti nad bazom mapiraju u ekvivalentne upite relacijske baze podataka.

I. Viševrijednosne baze podataka (eng. Multivalue Databases)

Poznatije su još i kao PICK baze podataka. Razlikuju se od relacijskih baza po tome što podržavaju atribute koji mogu imati više vrijednosti. Tablice u bazi su ekstremno fleksibilne i mogu sadržavati male programe za izračune vrijednosti stupaca.

J. Višemodelne baze podataka (eng. Multimodel Databases)

Razne kombinacije navedenih nerelacijskih baza podataka.

III. UPITNI JEZIK

Autori u [4] i [34] složili su se da trenutno stanje na tržištu upitnog jezika NoSQL baza podataka možemo usporediti sa erom prije Codd-ovog predstavljanja SQL jezika: zbog prekobrojnosti proizvoda NoSQL baza podataka ne postoji jedinstveni jezik za sve, već svaka baza podataka ima svoj karakterističan upitni jezik. Međutim, postoje i neka istraživanja u smjeru kreiranja unificiranog jezika za sve NoSQL baze podataka [35] [34], ili barem za pojedine kategorije.

U [34] autori navode kako NoSQL tržište zadovoljava 3 karakteristike monopolistički kompetitivnog tržišta: niski kriteriji za ulazak i izlazak konkurenata, veliki broj malih proizvođača te mnogobrojni heterogeni i jako različiti proizvodi, što naravno utječe na brojnost i različitost NoSQL upitnih jezika. U tome kontekstu autori su predstavili podlogu upitnog jezika naziva coSQL za unificiranje upitnog jezika ključ – vrijednost NoSQL baza podataka.

U [35] predstavljen je jezik naziva UnQL kao upitni jezik polustrukturiranih podataka predstavljenih grafom. Opisana je deklarativna sintaksa slična SQL upitnom jeziku koja koristi i select – where klauzulu, te je opisan model polustrukturiranih podataka baziranih na označenom grafu. UnQL jezik namijenjen je korištenju u dokument bazama podataka gdje se upiti vrše nad JSON dokumentima.

Obzirom da još uvijek nema jedinstvenog upitnog jezika za sve, upiti se nad bazom najčešće izvršavaju korištenjem specifičnih upitnih jezika pojedinih distribucija baza podataka preko komandne linije ili programskog sučelja (API-ja). U domeni web aplikacija vrlo je koristan REST (eng. *REpresentational State Transfer*) protokol preko kojeg se vrši komunikacija i prijenos podataka među komponentama distribuiranog sustava.

U slučaju stupčanih i dokument baza podataka obrada upita nad velikim skupom podataka pohranjenih na više računala je prilično zahtjevan posao samo jednom računalu. Zato se koristi Apache Hadoop programski paket otvorenog koda koji služi za distribuirano spremanje datoteka i distribuiranu obradu podataka. Hadoop daje mogućnost analize i obrade podataka pomoću Hadoop MapReduce sustava [7]. Razni programi za analizu podataka poput Pig [36] i Hive [37] sadrže izraze za agregiranje, filtriranje, ubacivanje, transformacije podataka te na taj način olakšavaju bolju protočnost podataka među sustavima. Navedeni programi se prevode u niz MapReduce poslova.

Kod graf baza podataka upite možemo vršiti nad čvorovima i vezama. Gremlin [38] je imperativni programski jezik za prolazak kroz graf korištenjem XPATH [1] upitnog jezika. Sparq [39] je popularan deklarativan jezik jednostavne sintakse. Koristi se za pronalazak uzoraka u grafu.

[1] XPath (engl. *XML Path language*) je upitni jezik za odabir čvorova iz XML dokumenta. Definiran je od strane World Wide Web Consortiuma (W3C)

Možemo reći da se NoSQL baze podataka međusobno jako razlikuju po funkcionalnostima upitnih jezika. Osim uzimanja u obzir model podataka i kako on utječe na postavljanje upita nad pojedinim atributima, potrebno je obratiti pažnju i na sučelja koja pojedini upitni jezici nude kako bi čim uspješnije pronašli bazu podataka za određeni slučaj.
Ako je potrebno jednostavno API sučelje, nevezano za neki konkretan programski jezik, najbolje je uzeti u obzir REST sučelja, posebno u domeni web aplikacija. Kod upita koji kritično utječu na performanse najbolje je primijeniti API sučelja dostupna za gotovo svaki zajednički jezik kao što je Java. Upitni jezici pojedinih baza podataka nude višu razinu apstrakcije kako bi se smanjila kompleksnost. Posebno su korisni kod izvršavanja složenijih upita. U slučaju upita intenzivnih računanja nad velikim skupom podataka, najbolje je koristiti MapReduce sučelje.

IV. SKLADIŠTA PODATAKA

Skladište podataka je baza podataka koja sadrži povijesne nepromjenjive podatke koji se prikupljaju i obrađuju radi potpore poslovnom odlučivanju [40]. Karakteristike skladišta podataka su tematska orijentiranost, konzistentnost i integriranost podataka, nepromjenjivost te prikaz vlastitog razvoja kroz vrijeme. Podaci se izvlače iz raznovrsnih izvora te se u skladu s definiranim modelom podataka učitavaju u skladište podataka i integriraju s postojećim podacima.

Prvotno su izvori podataka podrazumijevali relacijske baze sa numeričkim, integriranim i transakcijskim podacima. Povećanjem zahtjeva za analizu podataka korporacije uzimaju u obzir, ne samo korporativne podatke transakcijskih baza podataka, već sve podatke iz kojih se mogu izvući poslovne vrijednosti. To uključuje tekst i nestrukturirane podatke, uz numeričke i transakcije podatke. Time se, osim problema pohrane nestrukturiranih podataka (koji se nastoji riješiti NoSQL bazama podataka), pojavio i problem integracije tih podataka u skladišta podataka.

Područna skladišta podataka (eng. *Data Marts*) definiramo kao manji skup podataka dizajniran i konstruiran radi potpore odlučivanju pri čijem se dizajniranju slijede principi dizajna skladišta podataka, s time da taj skup podataka koristi homogena grupa korisnika [41]. Posebno su značajna jer služe kao platforma za OLAP analizu podataka (eng. *Online Analytical Processing*) te predstavljaju dio cjelovite arhitekture skladišta podataka.

ETL procesi (eng. *extract, transform and load*) predstavljaju procese izvlačenja, transformacije i punjenja podataka, iz različitih izvora podataka pa kroz sve slojeve arhitekture skladišta podataka [42]. To je skup koraka i aktivnosti potrebnih da bi se podaci iz izvora podataka unijeli u centralno skladište podataka te se iz centralnog skladišta podataka transformirali i unijeli u područna skladišta podataka. Danas na tržištu postoje brojni ETL alati.

Sustav poslovne inteligencija (eng. *Business intelligence*, BI) možemo opisati kao skup tehnologija i mehanizama za učinkovito izvlačenje poslovnih informacija iz velikog opsega podataka. Skladište podataka možemo shvatiti kao infrastrukturu za poslovnu inteligenciju, dok OLAP predstavlja alat kojim se ta analiza izvršava. Da bismo mogli izvoditi OLAP analizu, za pohranu podataka u skladište podataka koristi se višedimenzionalni model podataka [43]. Podaci su najčešće organizirani i pohranjeni prema zvjezdastoj shemi, gdje tablica činjenica pohranjuje mjerljive činjenice poslovnih događaja, a tablice dimenzija daju kontekst činjenicama. Takva shema omogućuje bolje performanse upita zbog redukcije join operatora.

Što se tiče arhitekture skladišta podataka, u današnje doba postoje brojne inačice. Ovdje ćemo spomenuti 3 osnova tipa arhitekture definirane prema [44], promatrane po broju slojeva, pa tako imamo: jednoslojnu, dvoslojnu i troslojnu arhitektura. Jednoslojna arhitektura se rijetko koristi a njena glavna zadaća je smanjiti obujam podataka uklanjanjem redundancije među podacima. Skladište podataka je virtualno, odnosno implementirano samo kao višedimenzionalni pogled operativnih podataka. Dvoslojna arhitektura dolazi u 2 oblika: sa uključenim slojem za režiranje i bez sloja za režiranje. Dvoslojna arhitektura sa slojem za režiranje sastoji se od izvora podataka sa jedne strane, sloja za režiranje i područnih skladišta podataka sa druge strane (eng. *Operational Data Store, ODS*). Dvoslojna arhitektura bez sloja za režiranje sa jedne strane ima izvore podataka dok sa druge područna skladišta podataka. Najčešće korištena je troslojna arhitektura u kojoj je centralno skladište podataka pozicionirano između više različitih izvora podataka i područnih skladišta podataka.

Dodatno, u [45] autori su predstavili novi standard arhitekture skladišta podataka sa naglaskom na praćenje promjena u podacima tokom vremena. Autori su prepoznali važnost životnog ciklusa podataka evidentiranog kroz dvije osnovne pojave: smanjena vjerojatnost pristupanja najstarijim podacima te povećanje količine učitanih podataka, što u konačnici dovodi do toga da povećanjem obujma podataka u skladištu podataka i veće količine starih podataka, smanjuje se postotak upotrebe tih podataka [46]. Naglasak su stavili na nestrukturirane podatke i na metapodatke koji imaju važnu ulogu u pružanju konteksta podacima pohranjenima u skladištu podataka te podacima generiranima upotrebom ETL alata. Metapodaci također mogu služiti za praćenje trenutnog i povijesnog stanja kroz cijelu infrastrukturu skladišta podataka [40].

Obzirom da navedena arhitektura uključuje i različite izvore podataka (uključujući i nestrukturirane izvore), javlja se problem integracije podataka iz različitih izvora u centralno skladište podataka.

Autori u [47] predstavili su integraciju podataka iz NoSQL i relacijske baze podataka u jedinstvenu, virtualnu bazu podataka. Definirana su i pravila prevođenja upitnih jezika različitih NoSQL baza podataka (stupčane i dokument baze podataka). Upiti se pomoću definiranih pravila prevode iz SQL jezika u specifični jezik NoSQL baze podataka (upiti usmjereni relacijskim bazama se ne prevode).

U kontekstu NoSQL baza podataka, autori u [48], [49], [50], [51] istražili su implementaciju NoSQL baza podataka i područnih skladišta podataka.

U [48] prikazano je istraživanje upotrebe NoSQL baza podataka kod implementacije OLAP sustava. U tu svrhu predložena su 2 NoSQL logička modela te skup pravila kojima se vrši transformacija iz višedimenzionalnog konceptualnog modela u logički NoSQL model.

Isti su autori u [49] predstavili proširenje zvjezdaste sheme SSB (eng. *Star Schema Benchmark*) koja podržava distribuiranu NoSQL bazu podataka uz relacijsku bazu podataka. Podržani su i razni formati podataka (csv, JSON, XML) te normalizirani i denormalizirani podaci.

U [50] predstavljeno je proširenje testa nad zvjezdastom shemom naziva CNSSB (eng. *Columnar NoSQL Star Schema Benchmark*) zasnovanog na stupčanom NoSQL skladištu podataka. Test je implementiran za rad nad Hbase stupčanom sustavu za upravljanje bazom podataka.

U [51] autori su istaknuli prednosti korištenja područnog skladišta podataka zasnovanog na nerelacijskoj (NoSQL) bazi podataka MongoDB i Clusterpoint DB .

U [52] autori su istražili procesiranje OLAP upita nad stupčanom bazom podataka pomoću MapReduce sučelja. Analiziran je utjecaj distribuiranih atributa stupčane HBase baze podataka nad performansama OLAP upita. Značajan utjecan na performanse izvođenja upita imao je fizički dizajn skladišta podataka, odnosno broj pristupljenim dimenzijama te količina podataka.

U [53] autori su predstavili inkrementalni pristup fizičkog dizajna skladišta podataka baziranog na Data Vault modelu podataka [54]. Pristup je temeljen na algoritmu za inkrementalno proširenje skladišta podataka dodavanjem skupa izvora podataka ili jedan po jedan. Algoritam koristi model metapodataka i pravila dizajna počevši sa transakcijskim izvorima podataka te koristeći veze između entiteta u transakcijskim sustavima i pravila za razvoj skladišta podataka, temeljenog na Data Vault metodi, automatizira fizički dizajn modela skladišta podataka. Najvažniji doprinos ovog rada je realizacija Data Vault sheme direktno iz shema sustava za upravljanje bazom podataka.

V. ANALIZA I RASPRAVA

NoSQL baze podataka predstavljaju trend u području pohrane velike količine strukturiranih i nestrukturiranih podataka. Moramo imati na umu da NoSQL rješenja nisu kreirana iz istih razloga kao i SQL; dok je SQL primarno namijenjen strukturiranim podacima i upravljanju transakcijama, NoSQL nastoji riješiti problem pohrane podataka i masivne skupove nestrukturiranih i polustrukturiranih podatka. U konačnici, jedni i drugi imaju svoje prednosti i nedostatke koji se mogu dalje istraživati.

Što se tiče samih NoSQL modela podataka, možemo ih objediniti u 4 osnovne kategorije (ključ-vrijednost, stupčane, dokument i graf baze podataka) te se dalje mogu istražiti karakteristike pojedinih baza podataka ovisno o potrebama i performansama. Ključ-vrijednost baze podataka prigodne su za izvršavanje upita nad podacima u što kraćem vremenu te služe kao temelj za druge modele podataka. Graf baze podataka pokazale su se pogodnima za vizualizaciju i pohranu podataka na način intuitivan korisniku u kontekstu u kojem su veze među podacima jednako važne kao i sami podaci. Dokument i stupčane baze podataka pokazale su se najboljim izborom u kontekstu skladišta podataka. Klasična arhitektura skladišta podataka zasniva se na višedimenzionalnoj kocki i zvjezdastoj shemi sa izvorima podataka u relacijskoj shemi.

Problem pohrane velike količine strukturiranih i nestrukturiranih podataka kod skladišta podataka možemo podijeliti na 2 potproblema: a) integracija i pohrana nestrukturiranog tipa podataka iz različitih izvora podataka te b) analiza podataka područnih skladišta podataka.

Kao rješenje problema analize podataka područnih skladišta podataka spominju se NoSQL baze podataka kao potpora sustavima za odlučivanje [48] u obliku ekstenzija zvjezdaste sheme [49], [50] ili pretvorbe višedimenzionalnog modela [48] područnog skladišta podataka. Osim promjena u shemi područnog skladišta podataka, mogu se pratiti i promjene koje utječu na brzinu izvođenja upita različitim načinima distribucije atributa kao što je to u [52]. Neki od tih pristupa i metoda mogu se još dodatno istražiti i poboljšati. Kao prostor za dodatno istraživanje ovdje vidimo proširenje modela i programskog rješenja te testova na skup nestrukturiranih podataka kao i razvoj novih logičkih NoSQL modela podataka koji se mogu koristiti za sustave za potporu odlučivanju. Također, moguće je i postojeća rješenja proširiti dodavanjem novih, složenijih upita za testiranje kako bi se identificiralo koji od njih imaju najviše prednosti kod NoSQL modela podataka. Osim navedenoga, prostor za daljnje istraživanje je i u automatizaciji dizajna centralnog skladišta podataka iz NoSQL izvora podataka.

Osim problema obrade podataka u skladištima, možemo se orijentirati i na pohranu odnosno integraciju podataka iz distribuiranih NoSQL baza podataka u centralno skladište podataka, definirano prema nekoj verziji troslojne arhitekture skladišta podataka. Tradicionalno se podaci učitavaju iz različitih relacijskih izvora te se integriraju u centralno relacijsko skladište podataka (definirano po raznim relacijskim shemama). Međutim, danas je sve više i nestrukturiranih (NoSQL) izvora koje je potrebno integrirati u centralno skladište podataka. Relacijska shema prilagođena 3NF nije pogodna za pohranu velike količine polustrukturiranih i nestrukturiranihi podataka zbog normaliziranog modela podataka i ACID svojstva (atomarnost, konzistentnost, izolacija i trajnost) koji imaju negativan utjecaj na performanse i dostupnost sustava u distribuiranom okruženju eksponencijalnog porasta podataka. Ovaj problem je vrlo slabo zastupljen u trenutnim istraživanjima.

Kao smjer za vlastito istraživanje prirodno se nameće detaljnije proučavanje područja NoSQL baza podataka i skladišta podataka te poboljšanje postojećih i/ili definiranje novih metoda i tehnika integracije podataka iz

različitih izvora podataka u skladište podataka, uključujući i praćenje promjena u podacima (tzv. problem evolucije skladišta podataka). Smjer budućeg istraživanja još nije formalno definiran, međutim u ovome trenutku vide se 3 moguće opcije (ne nužno međusobno isključive): a) definiranje pravila za prevođenje NoSQL u Data Vault skladište podataka, b) automatizacija procesa integracije podataka sa naglaskom na NoSQL izvore podataka te c) proširenje tradicionalnog sistemskog kataloga NoSQL izvorima podataka te integracija metapodataka za praćenje nestrukturiranih i strukturiranih izvora podataka i promjena u njima.

VI. ZAKLJUČAK

U ovom radu predstavljen je trend NoSQL baza podataka te usporedba sa relacijskim modelom podataka kao i njihovo korištenje u kontekstu skladišta podataka. Predstavljen je problem pohrane velike količine polustrukturiranih i nestukturiranih podataka za koji formalna relacijska shema nije dovoljna. Definirana su ACID svojstva relacijskih baza te BASE svojstva NoSQL baza podataka i CAP teorem.

Također, predstavljeni su različiti modeli pohranjivanja podataka u NoSQL bazu podataka (ukupno 10 modela), od kojih uzimamo 4 kao referentne: ključ-vrijednost, stupčani, dokument i graf baze podataka. Definirane su pojedinosti svakog od njih, uz primjere najpoznatijih distribucija baza podataka.

Opisana su i istraživanja upitnih jezika pojedinih kategorija NoSQL baza podataka. Zaključeno je da ne postoji unificirani jezik pojedinih kategorija, te se upiti vrše preko API sučelja i REST protokola.

Možemo zaključiti da su se NoSQL baze podataka pokazale kao izvrstan odabir u slučaju upravljanja velikom količinom podataka te u primjeni horizontalne skalabilnosti i visoke propusnosti, međutim još uvijek nije dovoljno zrela tehnologija koja bi mogla u potpunosti zamijeniti relacijske baze. Naglasak bi u budućnosti mogao biti na hibridnim modelima u kojima bi se relacijska baza nadopunila nekim prikladnim NoSQL bazama podataka.

U kontekstu skladišta podataka opisana je veza skladišta podataka sa NoSQL bazama podataka te je napravljen pregled istraživanja u području skladišta podataka i NoSQL baza podataka. Ovdje su istraživanja podijeljena na 2 problema: a) integracija i pohrane nestrukturiranog tipa podataka iz različitih izvora podataka te b) analiza podataka područnih skladišta podataka. Analizom dosadašnjih istraživanja možemo zaključiti da su autori napravili značajan iskorak u rješavanju problema analize podataka područnih skladišta podataka, međutim ovdje ima još prostora za daljnje istraživanje. Prostor za vlastito istraživanje vidimo u području integracije i pohrane nestrukturiranog tipa podataka iz različitih izvora podataka u centralno skladište podataka, prvenstveno kroz moguće poboljšanje postojećih i/ili definiranje novih metoda i tehnika integracije podataka, uključujući i praćenje promjena u podacima.

LITERATURA

[1] Strozzi Carlo, „NoSQL Relational Database Management System: Home Page“, 2010. [Na internetu]. Dostupno na: http://www.strozzi.it/cgi-bin/CSA/tw7/I/en_US/nosql/Home Page. [Pristupljeno: 27-kol-2016].

[2] Evans Eric, „Eric Evans' Weblog“, 2009. [Na internetu]. Dostupno na: http://blog.sym-link.com/2009/10/30/nosql_whats_in_a_name.html. [Pristupljeno: 27-kol-2016].

[3] Tomić Nela, „NoSQL tehnologije i primjene“, Sveučilište u Zagrebu, Prirodoslovno - matematički fakultet, 2016.

[4] R. Hecht i S. Jablonski, „NoSQL evaluation: A use case oriented survey“, *Proc. - 2011 Int. Conf. Cloud Serv. Comput. CSC 2011*, str. 336–341, 2011.

[5] M. Hanine, A. Bendarag, i O. Boutkhoum, „Data-Migration-Methodology-from-Relational-to-NoSQL-Databases“, *Int. J. od Comput. Electr. Autom. Control Inf. Eng.*, sv. 9, 2015.

[6] S. Ghemawat, H. Gobioff, i S.-T. Leung, „The Google file system“, *ACM SIGOPS Oper. Syst. Rev.*, sv. 37, izd. 5, str. 29, 2003.

[7] J. Dean i S. Ghemawat, „MapReduce: Simplified Data Processing on Large Clusters“, *Commun. ACM*, sv. 51, izd. 1, str. 107–113, 2008.

[8] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, i R. E. Gruber, „Bigtable“, *ACM Trans. Comput. Syst.*, sv. 26, izd. 2, str. 1–26, 2008.

[9] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, i W. Vogels, „Dynamo“, *ACM SIGOPS Oper. Syst. Rev.*, sv. 41, izd. 6, str. 205, 2007.

[10] D. Pritchett, „Base: an Acid Alternative“, *Queue*, sv. 6, izd. 3, str. 48–55, 2008.

[11] E. Brewer, „CAP twelve years later: How the ‚rules' have changed“, *Computer (Long. Beach. Calif).*, sv. 45, izd. 2, str. 23–29, 2012.

[12] Hurst Nathan, „Visual Guide to NoSQL Systems - Nathan Hurst's Blog“, 2010. [Na internetu]. Dostupno na: http://blog.nahurst.com/visual-guide-to-nosql-systems. [Pristupljeno: 28-kol-2016].

[13] R. Cattell, „Scalable SQL and NoSQL data stores“, *ACM SIGMOD Rec.*, sv. 39, izd. 4, str. 12, 2010.

[14] J. Pokorny, „NoSQL databases: a step to database scalability in Web environment“, u *Proceedings of the International Conference on Computer Science and Software Engineering - C3S2E '13*, 2013, sv. 9, izd. September, str. 14–22.

[15] Y. Huang i T. J. Luo, „NoSQL database: A scalable, availability, high performance storage for big data“, u *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014.

[16] K. Grolinger, W. a Higashino, A. Tiwari, i M. A. Capretz, „Data management in cloud environments: NoSQL and NewSQL data stores“, *J. Cloud Comput. Adv. Syst. Appl.*, sv. 2, str. 22, 2013.

[17] K. Hamzeh, F. Marios, Z. Saeed, Beigi-Mohammadi, R. Brian, S. Mark, G. Purwa, i L. Martin, „How do I choose the right NoSQL soulution? A comprehensive theoretical and experimental survey“, *Discret. Contin. Dyn. Syst.*, sv. X, izd. 0, str. 1–33, 2014.

[18] V. Sharma i M. Dave, „SQL and NoSQL Databases“, *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, sv. 2, izd. 8, str. 20–27, 2012.

[19] „Memcached“. [Na internetu]. Dostupno na: https://memcached.org/. [Pristupljeno: 17-lis-2016].

[20] „Redis“. [Na internetu]. Dostupno na: http://redis.io/. [Pristupljeno: 15-lis-2016].

[21] „Oracle BerkeleyDB“. [Na internetu]. Dostupno na: http://www.oracle.com/technetwork/database/database-technologies/berkeleydb/overview/index.html. [Pristupljeno: 15-lis-2016].

[22] „Voldemort“. [Na internetu]. Dostupno na:

http://www.project-voldemort.com/voldemort/. [Pristupljeno: 15-lis-2016].

[23] „Riak KV“. [Na internetu]. Dostupno na: http://basho.com/products/. [Pristupljeno: 16-lis-2016].

[24] Apache, „HBase – Apache HBase™ Home“. [Na internetu]. Dostupno na: http://hbase.apache.org/. [Pristupljeno: 02-stu-2016].

[25] „Hypertable“. [Na internetu]. Dostupno na: http://www.hypertable.org/. [Pristupljeno: 03-stu-2016].

[26] Apache, „Cassandra“. [Na internetu]. Dostupno na: http://cassandra.apache.org/. [Pristupljeno: 04-stu-2016].

[27] „JSON“. [Na internetu]. Dostupno na: http://json.org/. [Pristupljeno: 17-lis-2016].

[28] Apache, „CouchDB“. [Na internetu]. Dostupno na: http://couchdb.apache.org/. [Pristupljeno: 03-stu-2016].

[29] „MongoDB“. [Na internetu]. Dostupno na: https://www.mongodb.com/. [Pristupljeno: 03-stu-2016].

[30] Couchbase, „Why NoSQL?“, izd. October. str. 1–10, 2013.

[31] „Neo4j: The World’s Leading Graph Database“. [Na internetu]. Dostupno na: https://neo4j.com/. [Pristupljeno: 03-stu-2016].

[32] Ontotext Graph DB ™, „Graph Database“. [Na internetu]. Dostupno na: http://ontotext.com/products/graphdb/. [Pristupljeno: 03-stu-2016].

[33] N. Jatana, S. Puri, M. Ahuja, I. Kathuria, i D. Gosain, „A Survey and Comparison of Relational and Non-Relational Database“, *Int. J. Eng. Res. Technol.*, sv. 1, izd. 6, 2012.

[34] B. Y. E. Meijer i G. Bierman, „A Co-Relational Model of Data for Large Shared Data Banks“, sv. 54, izd. 4, str. 49–58, 2011.

[35] P. Buneman, M. Fernandez, i D. Suciu, „UnQL: a query language and algebra for semistructured data based on structural recursion“, *VLDB J.*, sv. 9, izd. 1, str. 76, 2000.

[36] Apache, „Pig“. [Na internetu]. Dostupno na: https://pig.apache.org/. [Pristupljeno: 17-lis-2016].

[37] Apache, „Hive TM“. [Na internetu]. Dostupno na: https://hive.apache.org/. [Pristupljeno: 17-lis-2016].

[38] Apache TinkerPop, „The Gremlin Graph Traversal Machine and Language“. [Na internetu]. Dostupno na: http://tinkerpop.apache.org/gremlin.html. [Pristupljeno: 17-lis-2016].

[39] W3C, „SPARQL 1.1 Query Language“. [Na internetu]. Dostupno na: https://www.w3.org/TR/sparql11-query/. [Pristupljeno: 17-lis-2016].

[40] W. H. Inmon, *Building the Data Warehouse*. Wiley Computer Publishing, 1992.

[41] D. Pintar, Z. Skočir, i B. Vrdoljak, „Oblikovanje skladišta podataka“, Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva, Zagreb.

[42] R. Kimball, *The Data Warehouse Lifecycle Toolkit*. New York: Wiley, 2008.

[43] R. Kimball, M. Ross, i J. Caserta, *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*. Wiley, 2004.

[44] M. Golfarelli i S. Rizzi, *Data Warehouse Design, Modern principles and methodologies*. The McGraw-Hill Companies, S.r.l.-Publishing Group Italia, 2009.

[45] W. H. Inmon, D. Strauss, i G. Neushloss, *DW 2.0: The Architecture for the Next Generation of Data Warehousing*. Morgan Kaufmann, 2010.

[46] B. H. Inmon, „Data Warehousing, 2.0 Modeling and Metadata Strategies for Next Generation Architectures“, *Forest Rim Technology, LLC*, 2010. .

[47] O. Curé, R. Hecht, C. Le Duc, i M. Lamolle, „Data Integration over NoSQL Stores Using Access Path Based Mappings“, *Proc. 22Nd Int. Conf. Database Expert Syst. Appl. - Vol. Part I*, sv. 1, izd. ii, str. 481–495, 2011.

[48] M. Chevalier, M. El Malki, A. Kopliku, O. Teste, i R. Tournier, „Implementing multidimensional data warehouses into NoSQL“, *ICEIS 2015 - 17th Int. Conf. Enterp. Inf. Syst. Proc.*, sv. 1, str. 172–183, 2015.

[49] M. Chevalier, M. El Malki, A. Kopliku, O. Teste, i R. Tournier, „Benchmark for OLAP on NoSQL technologies comparing NoSQL multidimensional data warehousing solutions“, *Res. Challenges Inf. Sci. (RCIS), 2015 IEEE 9th Int. Conf.*, sv. 3, str. 480–485, 2015.

[50] K. Dehdouh, O. Boussaid, i F. Bentayeb, „Columnar NoSQL Star Schema Benchmark“, *Proc. Model Data Eng. 4th Int. Conf. MEDI 2014*, str. 281–288, 2014.

[51] Z. Bicevska, A. Neimanis, i I. Oditis, „NoSQL-based Data Warehouse Solutions : Sense , Benefits and Prerequisites“, sv. 4, izd. 3, str. 597–606, 2016.

[52] L. C. Scabora, J. J. Brito, R. R. Ciferri, i C. D. De Aguiar Ciferri, „Physical data warehouse design on NoSQL databases: OLAP query processing over HBase“, *ICEIS 2016 - Proc. 18th Int. Conf. Enterp. Inf. Syst.*, sv. 1, 2016.

[53] D. Krneta, V. Jovanović, i Z. Marjanović, „A direct approach to physical Data Vault design“, *Comput. Sci. Inf. Syst.*, sv. 11, izd. 2, str. 569–599, 2014.

[54] D. Linstedt, *Super charge your data warehouse invaluable data modeling rules to implement your Data Vault*. Dan Linstedt, LearnDataVault.com, 2011.