UDC: 004.65 Info M: str. 4-10

PRIMENA VIRTUELIZACIJE U CILJU INTEGRACIJE RELACIONIH I NOSQL BAZA PODATAKA, NA PRIMERU ORACLE-A
USING VIRTUALIZATION FOR INTEGRATION OF RELATIONAL AND NOSQL DATABASES, IN THE CASE OF ORACLE

Srđa Bjeladinović, Zoran Marjanović
Fakultet organizacionih nauka, Univerzitet u Beogradu

REZIME: Pojava NoSQL baza podataka otvorila je pitanje njihove koegzistencije sa relacionim sistemima za upravljanje bazom podataka. Nakon početnih nesuglasica oko tumačenja nove paradigme, postalo je evidentno da će u narednom periodu ova dva pristupa u projektovanju struktura baza podataka paralelno postojati. Pitanja koje se u poslednjih nekoliko godina aktuelizovalo je da li je moguće napraviti mehanizam njihove integracije, pri čemu bi svako rešenje zadržalo svoje specifičnosti. Različiti pristupi su primenjivani u pokušaju davanja odgovora. Jedan od njih je i uvođenje sloja virtuelizacije. Ovaj pristup preuzet je iz referentnih radova (testirano na MySql i MongoDB), a u ovom radu je analizirana mogućnost proširenja skupa različitih baza nad kojima je moguće integrisati rešenje, konkretno sa Oracle-om. Pored testiranja mogućnosti integracije izvršeno je testiranje i analiza uticaja primene sloja virtuelizacije na performanse spomenutog sistema.

KLJUČNE REČI: NoSQL baze podataka, integracija, virtuelizacija, performanse

ABSTRACT: The emergence of NoSQL databases has opened the question of their coexistence with relational database management systems. After initial misunderstandings about the way new paradigm should be understand, it became evident that in the upcoming years, these two approaches (relational and NoSQL systems) will be used in parallel. In recent years, question whether is it possible to make the mechanism of their integration, in which any solution will retain its specifity became actualized. Different approaches were applied in attempt to give answer to the mentioned question. One of them is the introduction of the virtualization layer. This approach has been taken from reference works (tested on MySQL and MongoDB). This paper analyzes the possibility of expanding the set of database management systems that could be used as a data source for integration between relational and NoSQL systems, on the example of Oracle. Beside testing integration possibilities, impact on performance, after applying virtualization layer, was tested and analized.

KEY WORDS: NoSQL databases, integration, virtualization, peformance

1. UVOD

Relacioni model predstavljen je 1970. godine od strane Edgarda Codd-a, u to vreme istraživača IBM laboratorije, u San Jose-u, Kaliforniji. U radu objavljenom juna navedene godine[1], autor je predstavio relacioni model kao osnovu za kreiranje strukture baze podataka. Početkom sedamdesetih godina prošlog veka pojavili su se i prva komercijalna rešenja zasnovana na relacionom modelu. Tokom decenija koje su usledile, ovaj model ustoličio se kao dominantna paradigma na kojoj se zasnivaju sistemi za upravljanje bazama podataka, a neretko kao i jedina prihvatljiva alternativa za sistem kojem konstantno pristupa veliki broj klijenata [2]. U spomenutom radu, Codd uspostavlja osnove relacionog modela, definiše koncept n-torke kao osnovne jedinice za čuvanje podataka, objašnjava pojam normalizacije relacija i predstavlja operacije za rad nad relacionim modelom.

Pred kraj prošlog veka, a posebno jačanjem objektno-orijentisanog pristupa u programiranju javlja se ideja kreiranja novog načina organizacije strukture baza podataka. Jedan od pokušaja predstavlja kreiranje objektnih baza, koje je trebalo da imaju intuitivniju primenu u povezivanju i korišćenju sa aplikacijama razvijenim korišćenjem objektnih programskih jezika. Logična ideja, koja je u nekom trenutku obećavala da će uspeti da parira relaciono zasnovanim bazama podataka, u praksi je zaživela samo u ograničenoj meri. Iako je bilo i drugih pokušaja kreiranja sistema koji bi zamenili relacione sisteme, poput XML baza, te tehnologije nisu bile prihvaćene od strane korisnika, niti su ostvarile srazmerni tržišni udeo poput relacionih sistema[2]. Razlog tome nalazi se u činjenici da većina savremenih sistema i dalje ima potrebu za jednostavnošću, pouzdanošću i efikasnošću relacionih baza podataka, kriterijumima kojima spomenuti alternativni pristupi u kreiranju strukture baza podataka (objektne, XML) nisu uspele da odgovore na adekvatan način. Epilog je da su spomenute alternative uglavnom integrisane u relacione sisteme, npr. objektno-relacione baze podataka.

Početkom veka, menja se situacija na svetskom poslovnom tržištu, te ekspanziju doživljavaju one kompanije koje svoje poslovanje baziraju na ponudi novih proizvoda i servisa ili na drastičnom unapređenju postojećih. Za ove kompanije je zajedničko da su u velikoj meri počele da koriste sve pogodnosti korišćenja Interneta, direktnog povezivanja pružaoca i korisnika usluga, kao i aktiviranjem potrebe korisnika da pripadaju i sudeluju u radu sve popularnijih društvenih mreža. Kao primeri izdvajaju se Google, koji je unapredio pretragu i takoreći potisnuo sa tržišta sve "starije" engine za pretragu, ili Facebook-a koji je u dobroj meri preuzeo ideju MySpace-a, pojednostavio korišćenje i učinio taj servis dostupan širokom broju korisnika. U spomenutu kategoriju lako se mogu svrstati i kompanije poput Amazon-a, Twitter-a, LinkedIn-a, povezivanje ljudi putem "društvenih" mreža i dr. Iako je i ranije bio očigledan trend eksponencijalnog rasta količine novokreiranih

podataka, navedene kompanije svojom ekspanzijom doprinele su dodatnom naglašavanju potrebe brzog manipulisanja velikom količinom podataka, skladištenja velikog broja podataka, upotrebe podatka složene strukture, kao i njihovog međusobnog povezivanja (Perdue T. 2012).

S' obzirom na navedeno ni ne čudi što su spomenute kompanije bile začetnici novog pristupa u projektovanju baza podataka. Taj pristup je nazvan NoSQL. Na samom početku, akronim je izazvao konfuziju među korisnicima, a karateristična je bila i pojava polemike po velikom broju blogova šta zapravo podrazumeva, u to vreme, nova paradigma NoSQL. U početku, tumačena je u velikoj meri kao "NonSQL", tj. "No to SQL" i bila je tretirana kao prekretnica u korišćenju relacionih sistema. Ipak, ubrzo je postalo jasno da pokret NoSQL baza predstavlja težnju kreiranja sistema fleksibilnije strukture, koji bi efikasnije radili sa velikom količinom nestruktuiranih podataka u odnosu na relacione sisteme (čija jedna od glavnih karakteristika upravo predstavlja dobro formalizovana struktura podataka), bez pokušaja da se u potpunosti zamene relacioni sistemi. Tako se došlo do tumačenja akronima NoSQL-a kao "Not only SQL" [4].

Spomenute kompanije kreirala su svoja NoSQL rešenja, za specifičnosti njihovih potreba, a najpoznatija su: Google-ov BigTable, Amazon-ov Dynamo i Facebook-ova Cassandra. Ovi sistemi za upravljanje bazom podataka kao prioritet postavljaju performanse i brzinu odziva, na račun nekonzistentnosti podatka koja može nastati u određenim trenucima.

Glavna pitanja na koji ovaj rad treba da odgovori su:

- Da li je moguće proširenje predloženog rešenja koje koristi sloj virtuelizacije?
- Kakav je uticaj na performanse novointegrisanog sistema?
- Koji bi bili pravci daljeg unapređenja sistema?

U drugom delu rada izvršena je analiza referentne literature. Prikazani su radovi koji se bave problemom integracije relacionih i NoSQL sistema, uz objašnjenje cilja rada, ciljne grupe i veze sa ovim radom.

U trećem delu ovog rada dat je kratak teorijski osvrt na osnovne karakteristke NoSQL sistema i njihove različitosti u odnosu na relacione sisteme, a iz čega proističu specifičnosti u potencijalnoj integraciji dva tipa sistema.

U četvrtom delu ovog rada pokazano je kako se jedan od pristupa integraciji primenom virtuelizacije može proširiti i na novi sistem. U radu je za proširivanje istraživanja autora Lawrence-a [5] korišćen UnityJDBC softver za integraciju. Autori su izvršili testiranje sa MySQL bazom podataka, dok je u ovom radu proširena integracija i sa Oracle-om. Izvršeno je direktno testiranje upita i testiranje istih upita kroz UnityJDBC, analizirani su rezultati i mogućnosti daljeg unapređenja integracije, uvođenjem novih funkcionalnosti i optimizacijom performansi.

U petom delu rada izložena su zaključna razmatranja i dati su pravci daljeg istraživanja.

2. PREGLED OBLASTI

U ovom poglavlju analizirani su neki od relevantnih radova vezani za integraciju SQL i NoSQL sistema za upravljanje bazom podataka.

Svi analizirani radovi ne pružaju predloge načina rešavanja problema integracije. Neki od njih se bave problemom integracije na opštijem nivou [10][11], dok neki analiziraju mogućnosti kreiranja integralnih upitnih jezika [12][13]. Iako mogu biti od interesa za neko od daljih istraživanja ovde su detaljnije analizirani sistemi koji su direktnije povezani sa domenom istraživanja ovog rada.

Radi lakšeg prikaza, korišćen je tabelarni prikaz, a za svaki rad je utvrđen cilj, kratak sažetak, ciljna grupa, kao i povezanost sa temom ovog rada. Podaci su prikazani u Tabeli 1.

Tabela 1. – *Prikaz referentnih radova iz oblasti*

Lawrence [5]	
Cilj:	Predlog rešenja problema integracije između SQL i NoSQL sistema za upravljanje bazom podataka, uvođenjem sloja virtuelizacije.
Kratak sažetak:	Kratak pregled alternativnim pristupima projektovanja baze podataka, identifikacija osnovnih tipova NoSQL baza podataka i kratka analiza dosadašnjih pokušaja integracije. Uvođenje sloja virtuelizacije i testiranje rada nad MySQL i MongoDB sistemima.
Ciljna grupa:	Deo naučne zajednice zainteresovan za integraciju relacionih i NoSQL sistema koji ima tendenciju objedinjavanja SQL upitnog jezika sa upitnim jezicima NoSQL baza podataka.
Povezanost sa temom:	Postoji direktna veza sa temom ovog rada, jer je navedeni rad poslužio kao osnov istraživanja. U ovom radu je preuzeto arhitekturalno proširenje koje je predložio navedeni autor, testirana je mogućnost proširenja skupa sistema koji bi se mogli integrisati i analizirana je mogućnost primene na Oracle, kao i uticaj na performanse izvršavanja upita. Na osnovu izvršene analize, dat je predlog mogućeg poboljšanja.
Potey et al. [6]	
Cilj:	Dati teorijski okvir za konverziju struktuiranih sistema za upravljanje bazama podataka u nestruktuirane
Kratak sažetak:	U početnom delu rada analizirani su relevantni radovi iz oblasti, čija je tema klasifikacija različitih sistema od interesa (NoSQL sistema), mogućnost konverzije relacionih i NoSQL sistema kodiranjem aplikacija za konkretne sisteme i kreiranjem pogleda. Ostatak rada predstavlja teorijski opis koraka koje bi klijenti trebalo da preduzmu, u rešenju čiji status i stepen trenutne implementacije nije jasno vidljiv u radu, radi postizanja konverzije.
Ciljna grupa:	Čitaoci početnici u ovoj oblasti, koji žele da spoznaju osnovne izazove integracije i migracije Sql i NoSQL sistema
Povezanost sa temom:	Korišćen je isključivo informativno za uvođenje u oblast istraživanja i za analizu jednog od mogućih pristupa rešavanju problema od interesa i ovog rada. Isključivo teorijski rad, ne preterano relevantan za praktičan deo ovog istraživanja.

Tabela 1. – *Prikaz referentnih radova iz oblasti (nastavak)*

Alomari et al. [7]	
Cilj:	Predlog rešenja problema integracije između različitih NoSQL sistema sa podrškom za integraciju i sa SQL sistemima
Kratak sažetak:	Detaljan opis problema i pregled referentne literature. U radu je prikazan cloud framework razvijen od strane autora, koji omogućuje povezivanja NoSQL i SQL sistema. Glavna komponenta je sloj između API-a sistema i sloja podataka u kojem se vrši prilagođavanje upita. Nakon prijema upita, spomenuti sloj vrši ponovo kreiranje upita, u skladu sa odredišnom bazom, a putem specijalizovanog adaptera i tako omogućava apstrakciju upitnih jezika i korišćenje i SQL upita za NoSQL sisteme. Eskperimentalno su pokazali u radu da upotreba framework-a zahteva minimalno kodiranja u cilju postizanja integracije.
Ciljna grupa:	Deo naučne zajednice zainteresovan za integraciju relacionih i NoSQL sistema koji ima tendenciju objedinjavanja SQL upitnog jezika sa upitnim jezicima NoSQL baza podataka.
Povezanost sa temom:	Spomenuti rad je direktno povezan sa ovim radom, jer analizira isti problem i daje rešenje koje je slično rešenju koje ovaj rad proširuje [5]. Detaljnije analizirano rešenje, sa prikazanim mehanizmima evaluacije, moglo bi biti iskorišćeno u daljem radu i pri poboljšavanju rešenja prikazanom u ovom radu.
Wu et al. [8]	
Cilj:	Analiza pogodnosti korišćenja SQL i NoSQL sistema u odgovarajućim scenarijima, oblasti primenjivosti i poređenje njihovih performansi
Kratak sažetak:	Rad u osnovi predstavlja detaljno poređenje između SQL i NoSQL sistema, uz kreiranje slučajeva korišćenja i testiranje performansi u istima.
Ciljna grupa:	Deo naučne i poslovne zajednice zainteresovan za poređenje performansi i domena primenjivosti SQL i NoSQL sistema.
Povezanost sa temom:	Predstavljene su sličnosti i razlike između SQL i NoSQL sistema, njihovog funkcionisanja u odgovarajućim slučajevima korišćenja, kao i adekvatnost njihove primene u zavisnosti od zahteva, čime je stvorena dobra osnova za identifikaciju domena "preklapajuće" primene SQL-a i NoSQL-a, odnosno domena u kojima bi trebalo koristiti isključivo jedan od dva tipa sistema (SQL ili NoSQL). Rad je posebno interesantan zbog performansi MongoDB-a, korišćenog i u referentnom radu [5].
Roijackers et al. [9]	
Cilj:	Integracija relacionih i NoSQL sistema (konkretno document-oriented baza, kao predstavnika NoSQL sistema) i predlog definisanja načina prevazilaženja razlika, radi bržeg izveštavanja u sistemima koji koriste BigData (istovremeno nad relacionim i document-oriented bazama podataka).
Kratak sažetak:	U radu je predstavljen framework za rešavanje opisanog problema povezivanja. Struktura dokumenata NoSQL sistema je predstavljena relacionim modelom, a SQL je proširen ekstenzijom novog, prilagođenog NoSQL upitnog jezika, razvijenog od strane autora. SQL i NoSQL se povezuju kroz odgovarajuće binding-e, dok je korisnicima ostavljena mogućnost opisivanja uslova pretrage nad dokumentima.
Ciljna grupa:	Deo naučne zajednice koji ima potrebu korišćenja i unapređenja sistema koji koriste BigData nad relacionim i NoSQL bazama podataka
Povezanost sa temom:	Rad rešava problem od interesa i za ovo istraživanje. Domen rešenja je sužen na document-oriented NoSQL baze podataka, ali je pristup rešavanja problema uporediv sa rešenjem iz referentnog rada[5]. Takođe, rad sadrži empirijske rezultate testiranog rešenja na PostgreSQL i MongoDB (koja je takođe testirana i u spomenutom referentnom radu).

3. OSNOVNE KARAKTERISTIKE NOSQL SISTEMA

SQL (Structured Query Language) baze podataka skoro pet decenija predstavljaju jedan od primarnih načina za skladištenje i obradu podataka, gde su posebno dobile na značaju početkom ove dekade sa sve većim razvojem i primenom web aplikacija i open-source rešenja, kao što su MySQL, PostgreSQL i SQLite. Nasuprot tome, NoSQL baze podataka, koje se zasnivaju na posebnim mehanizmima za čuvanje i rad nad podacima koji nisu modelovani na tradicionalan tabelarni način, poslednjih deset godina dobijaju značajno mesto na IT sceni pre svega sa razvojem i nastankom specijalizovanih baza velikih kompanija: MongoDB, CouchDB, Redis i Apache Cassandra. Iako se prvi put u literaturi pojavljuju 1960. godine, najveći broj tipova NoSQL baza se pojavio u poslednjih nekoliko godina. U velikoj meri su inspirisani Amazon-ovim Dynamo-om, koji svoje funkcionisanje bazira na hash tabeli. Sama hash tabela se može posmatrati kao posebna struktura podataka koja koristi hash funkciju za mapiranje identifikovanog ključa sa njemu dodeljenim vrednostima, te se tako putem određenog ključa pristupa konkretnim podacima, čime je eliminisana potreba pretrage celog skupa podataka. Navedena karakteristika znatno utiče na poboljšanje performansi i čini jednu od krucijalnih razlika u odnosu na relaciono struktuirane baze.

Sagledavajući osnovne funkcionalnosti i karakteristike koje predstavljaju zajedničku karakteristiku svih rešenja zasnovanih na NoSQL bazama podataka, kao centralni činioc se može izdvojiti CAP teorema, profesora Berkley univerziteta, dr. Eric-a Brewer-a [14]. Osnovu CAP teoreme predstavlja upravo akronim reči Consistency (atomnost), Availability (dostupnost) i tolerance to network Partitions (tolerancija ka deljenju mreže). Sama teorema glasi: "Svaki distribuirani sistem može zadovoljiti najviše dve od sledeće tri karakteristike: atomnost, dostupnost i toleranciju ka deljenju mreže". CAP teorema je dokazana 2012.godine na MIT univerzitetu od strane Seth Gilbert i Nancy Lynch [15], gde je dokazano da jedino potpuni prestanak rada mreže (partition tolerance) može doprineti da sistem funkcioniše nekorektno, gde servis koji raspolaže sa consistency funkcioniše u potpunosti ili ne funkcioniše uopšte, dok ukoliko na svaki zahtev prihvaćen od strane ispravnog čvora usledi odgovor sistema, distribuirani sistem se smatra dostupnim.

Kao jedna od osnovnih razlika u odnosu na relacione baze podataka i SQL, koje se baziraju na ACID osobinama (Atomicity (Atomnost), Consistent (Konzistentnost), Isolation (Izolacija), Durability (Trajnost)), za NoSQL baze podataka važe BASE osobine, putem kojih se marginalizuje značaj konzi-

stentnosti i izlolovanosti iz ACID osobina u korist dostupnosti, "graciozne degradacije" i performansi [14]. Koncept BASE osobina je sačinjen od sledećih karakteristika:

- Basically Available – osnovna dostupnost sistema se zasniva na mogućnosti da nije neophodno da svi delovi sistema moraju da budu dostupni u svakom trenutku, da bi on bio dostupan.
- Soft-state –Konzistentnost sistema nije neophodna u svakom trenutku.
- Eventual consistency – eventualna konzistentnost se zasniva na postulatu da će biti situacija kada će sistem biti u konzistentnom stanju, dok u nekom konkretnom trenutku to ne mora biti slučaj. Nakon nekog vremena, svi čvorovi će se ponovo vratiti u konzistentno stanje.

Pored BASE osobine, koje su svakako krucijalne za funkcionisanje NoSQL baza podataka, kao takođe važna karakteristika, izdvaja se skalabailnost sistema, koja se definiše kao odlika sistema, mreže ili procesa da odgovori na rastuću količinu posla na adekvatan način [16]. NoSQL baze podataka podržavaju horizontalnu skalabilnost, koja omogućava dodavanje novih čvorova sistemu, npr. prelazak sa jednog web servera na sistem stabla servera.

Razlikovanje i kategorizacija NoSQL baza podataka može se definisati shodno različitim načinima skladištenja samih podataka. U prvoj fazi razvoja i primene NoSQL baza podataka bila su dominantna dva rešenja, BigTable Google-a i Dynamo Amazona, koja su ujedno predstavljala i dve osnovne kategorije. Daljim razvojem i unapređenjem različitih kategorija baza podataka, danas, prema dostupnoj literaturi, može se izdvojiti 5 osnovnih kategorija [17]:

- key/value baze,
- baze zasnovane na koloni,
- baze zasnovane na dokumentu,
- baze grafovi,
- objektne baze.

NoSQL baze podataka koje se zasnivaju na principu key/value skladištenja predstavljaju najzastupljenije rešenje, a bazirano je na jednostavnom modelu koji čini mapa/direktorijum koji omogućava korisnicima da postavljaju i traže oređene vrednosti ključa. Osim modela podataka i API-a, ovaj tip NoSQL baze favorizuje veliku skalabilnost na račun konzistentnosti i zbog toga najveći broj rešenja iz ove kategorije ne podržava ni složene ad-hoc upite, ni analitičke funkcionalnosti, gde je dužina ključa najčešće ograničena brojem bajtova. Visoka dostupnost sistema generalno, pa tako i NoSQL sistema je tema koja zavređuje pažnju i mogu se naći interesantni radovi na ovu temu [18][19]. Pored dostupnosti, oblast koja zavređuje posebnu pažnju je svakako sigurnost sistema, pa tako i NoSQL sistema.

Tipično karakteristika NoSQL sistema je da su oni dizajnirani bez pridavanja posebnog značaja aspektu signosti, čime je korisnicima ostavljen prostor, ali i obaveza uvođenja sigunosnih tehnika i mehanizama [20]. U literaturi se mogu pronaći različiti modeli zaštite informacija [21], a u nastavku će biti prikazani najčešći sigurnosni izazovi sa kojima se susreću korisnici različitih sistema, pa tako i NoSQL sistema.

Za prikaz nekih od identifikovanih sigurnosnih aspekata biće korišćeni MongoDB i Cassandra, a na osnovu detaljno urađene analize [22]:

- Komunikacija između klijenta i čvora – MongoDB ne podržava enkripciju, dok Cassandra po defaultu nema uključenu opciju enkriptovanja. Ipak, Cassandra omogućava naknadno uvođenje SSL-a i to sa sledećim opcijama: ALL (sa svim unutrašnjim čvorovima), DC (enkripcija samo sa datacentrima) i RACK (enkripcija samo sa rekovima)
- Komunikacija između više čvorova – Iste opcije i podešavanja kao u komunikaciji između klijenta i čvorova
- Autorizacija – MongoDB koristi samo predefinisane uloge za baze podataka
- Praćenje (audit) – MongoDB ima malo opcija i uglavnom se tiču pračenja performansi, a ne pristupa i korišćenja različitih resursa, dok su kod Cassandre dostupne u Enterprise-u
- Autentifikacija – MongoDB po default-u ima isključenu opciju. Korisnici baze podataka su limitirani samo na nivo 1 logičke baze. Moguća je dodatna primena Kerberosa. Cassandra je po default-u svima dostupna, tj. nema autentifikacije. Šifre korisnika se heširaju i takođe podržava dodatnu primenu Kerberosa.

Sve opisano potvrđuje već spomenuto zapažanje [20] da NoSQL nisu kreirane sa idejom maksimizacije zaštite podataka i pristupa istima. Neki apsekti nisu gotovo uopše obuhvaćeni (MongoDB nema direktnu podršku kriptovanju), dok su neke dostupne opcije po default-u isključene (npr. autentifikacija u Cassandri). Ovo bi trebalo imati na umu pre početka korišćenja NoSQL sistema i definitivno bi trebalo napraviti specifikaciju minimalnog obima tehnika zaštite koje je potrebno uvesti pre populisanja NoSQL sistema podacima.

Ubrzani razvoj informaciono-komunikacionih tehnologija, sa posebnim akcentom na Internet, neprestano se povećava, a samim time i količina podataka koju je potrebno prikupiti, analizirati, obraditi i proslediti. Neophodnost u poslovanju, ali i u svakodnevnom životu, bazira se na utvrđivanju i definisanju pitanja performansi, pouzdanosti, dostupnosti, konzistentnosti i trajnosti podataka, ali i dostupnost i tolerancija ka deljenju mreže, koji su značajniji od striktne atomnosti, pogotovu kod velikih web aplikacija. U takvoj situaciji NoSQL baze podataka sa primenom CAP teoreme, koja dozvoljava povezivanje samo dve od tri osobine: atomnost, dostupnost i toleranciju ka deljenju mreže, omogućava njihovu (web aplikacija i sadržaja) bolju dostupnost i redundantnost, sa posebnim akcentom na primenu pristupa BASE osobina. Sve opisane karakteristike utiču na specifičnosti upita koje se koriste za rad sa podacima NoSQL sistema, što onemogućava direktno korišćenje SQL upitnog jezika. Usled toga, trenutno jedina prihvatljiva opcija, dok se eventualno ne kreira zajednički upitni jezik, predstavlja integracija sistema, pri čemu se svakom sistemu pristupa uz pomoć odgovarajućeg jezika. Naravno, glavna težnja je da korisnik ne bude upućen u to kojem sistemu i na koji način se pristupa, da ne razmišlja "kako" će nešto biti urađeno, već samo "šta" će biti urađeno. Korak ka tom cilju bi trebalo da omoguće i eksperimentalni rezultati dobijeni u ovom radu

(koji bi trebalo da doprinesu daljem razvoju ovog pristupa), a koji su predstavljeni u narednom delu rada.

4. EKSPERIMENTALNO TESTIRANJE

Merenje performansi sistema zahteva temeljni pristup. Mnogo je faktora, tj. varijabli koje mogu uticati na tačnost i sveobuhvatnost merenja. U ovom delu rada opisano je izvršeno kvantitativno testiranje. Testiranje obuhvata samo jedan aspekt performansi sistema: brzinu izvršavanja naredbi.

4.1. *Postavka testa*

Za testiranje su izabrane sve četiri osnovne DML (Data Manipulation Language) naredbe SQL-a: SELECT, DML naredba koja se najčešće koristi, INSERT naredba koja često zna da bude izvor zagušenja sistema, kao i UPDATE i DELETE, naredbe za koje je tipično očekivano brže izvršavanje u odnosu na dve prethodno navedene. Poređeno je izvršavanje upita direktno na Oracle EE 11g Release 2 bazi i na istoj bazi kojoj se pristupa putem UnityJDBC-a. Tokom testiranja INSERT naredbe generisani su podaci i to u tri iteracije: 1.000, 10.000 i 100.000 n-torki po pozivu. SELECT naredba je zatim izvršavana nad istim podacima. Svi upiti su testirani nad istim računarom (Intel i5 2GHz, 4GB RAM, 64-bitni Windows), a memorija je nakon svakog izvršavanja pražnjenja (relevantno za eliminaciju keširanja pri uzastopnom izvršavanju SELECT naredbe). Upiti sadrže manipulaciju predefinisanim tipovima podataka: karakternim, numeričkim i datumskim.

4.2. *Prikaz i analiza rezultata*

Testiranje spomenutih naredbi je izvršeno u serijama. Svaka naredba je ponovljena 50 puta, minimalna i maksimalna vrednost svakog testiranja je uklonjena, a prikazan rezultat predstavlja prosečnu vrednost 48 izmerenih vrednosti. Za računanje vremena korišćeni su predefinisani alati za prikaz vremena izvršavanja dostupni u okviru Oracle SQL Developer-a, kao i UnityJDBC-a. Izmerena vremena prikazana su u narednoj tabeli.

Tabela 2: – *Izvršavanje INSERT, SELECT, UPDATE i DELETE naredbi (prosečno vreme izvršavanja u sekundama, dobijeno iz 48 merenja)*

n-torki	1.000	10.000	100.000
Oracle 11g R2 EE			
INSERT	0,46	3,175	13,389
SELECT	0,71	2,947	26,35
UPDATE	0,05	0,11	2,503
DELETE	0,12	0,421	4,276
UnityJDBC + Oracle 11g R2 EE			
INSERT sa SELECT-om	(0,29)	(0,52)	(2,325)
SELECT	2,938	4,314	27,996
UPDATE	0,14	0,156	6,57
DELETE	0,4	0,591	11,61

Iz Tabele 2 se vidi da je u svim slučajevima izvršavanje upita preko SQL Developer-a bilo brže. To je bilo i očekivano, jer se pristupa direktno bazi i "zaobilazi se" dodatni sloj virtuelizacije, što utiče na performanse.

Ukoliko se posmatra izvršavanje naredbi SELECT, UPDATE i DELETE nad setom od 1.000 redova lako se može uočiti da je svaka naredba u proseku imala 3 do 4 puta duže izvršavanje prilikom pokretanja putem UnityJDBC-a, nego direktnim izvršavanjem nad relacionom bazom. Konkretno, izvršavanje SELECT naredbe sa spomenutim setom podataka trajalo je 0,71 sekundu u odnosu na 2,938 u slučaju kada je ista naredba pozvana iz UnityJDBC-a. Analogno tome, UPDATE naredbi je bilo potrebno 0,05 sekundi direktno nad Oracle bazom, odnosno 0,14 sekundi putem UnityJDBC-a, dok je naredba DELETE trajala 0,12 sekundi nad Oracle-om. odnosno 0,4 putem UnityJDBC-a.

Sa povećanjem broja n-torki uočava se trend smanjenja razlike između trajanja izvršavanja naredbi pozvanih direktno nad Oracle bazom i onih izvršenih kroz UnityJDBC i korišćenjem sloja virtuelizacije. Objašnjenje se može pronaći u činjenici da pristup sloju virtuelizacije predstavlja slabo promenljiv trošak, tako reći fiksan dodatni utrošak vremena u odnosu na direktan pristup bazi podataka. Sa povećanjem broja n-torki taj "fiksni" dodatni utrošak se "raspoređuje" na veći broj n-torki, čime se smanjuje razlika između prosečnog vrmenea izvršavanja naredbi, putem dva objašnjena pristupa. Za slučaj izvršavanja SELECT, UPDATE i DELETE naredbi sa 10.000 redova postignuti su sledeći rezultati respektivno direktno nad Oracle-om: 2,947, 0,11 i 0,421 sekundi, dok su iste naredbe putem UnityJDBC-a bile izvršavane u proseku za: 4,314, 0,156 i 0,591 sekundi.

Prilikom izvršavanja naredbi sa 100.00 redova, razlika između direktnog pristupa i pristupa kroz sloj virtuelizacije je dodatno smanjena, zbog objašnjenog razloga, pa su SELECT, UPDATE i DELETE naredbe postigle sledeće rezultate: direktnim pristupom Oracle-u prosečne izmerene vrednosti su bile 26,35, 2,503 i 4,276, dok su za korišćeni UnityJDBC vrednosti bile 27,996, 6,57 i 11,61 sekundi.

Kada je reč o INSERT naredbi, uočeno je jedno bitno ograničenje UnityJDBC-a: proceduralno izvršavanje SQL naredbi nije podržano, tj. u slučaju Oracle-a ne postoji podrška za PL/SQL. Ovim je izvršavanje INSERT naredbe poprilično otežano, u odnosu na direktan pristup Oracle bazi, gde, naravno, postoji mogućnost korišćenja PL/SQL-a. Spomenuti tehnički nedostatak dobija na značaju ukoliko se obrati pažnja na broj n-torki koji je korišćen za testiranje u ovom radu. Naime, INSERT 100.000 redova bez podrške konstrukcija poput petlji, bulk insert-a i slično onemogućava da se ravnopravno uporede relacioni i NoSQL sistem, upotrebom UnityJDBC-a. Navedeno ograničenje je uslovilo da se umesto klasičnog generisanja i unosa podataka pomoću INSERT naredbe iskoristi INSERT SELECT kombinacija. Tako upotrebljena INSERT naredba proizvodi efekat kopiranja redova iz jedne u drugu tabelu, čime se ne vrši generisanje vrednosti, te je i očekivana brzina izvršavanja naredbe manja od brzine izvršavanja uobičajene INSERT naredbe. To su pokazali i dobijeni rezultati: 0,29, 0,52

i 2,325 su prosečna vremena izvršavanja naredbe INSERT SELECT u sekundama sa 1.000, 10.000 i 100.000 redova respektivno. Direktan unos u Oracle 1.000, 10.000 i 100.000 redova prosečno je trajao 0,46, 3,175 i 13,389 sekundi respektivno.

Iz prikazanih rezultata može se izvesti više zaključaka. Razlike u vremenu izvršavanja postoje, ali se u zavisnosti sa brojem testiranih n-torki, a posebno u slučaju čitanja 100.000 n-torki, mogu smatrati zanemarljivim sa aspekta celokupnih performansi. Naravno, očekivano je bilo da direktan pristup postigne bolje rezultate, od pristupa korišćenjem "posrednika" (UnityJDBC-a).

Kada se uporedi sa rezultatima postignutim za predstavnika NoSQL baze, MongoDB, uočava se sličnost. Prosečno odstupanje SELECT naredbe je bilo oko 2%, ali je razlika sa INSERT naredbom bila 30%. To se pre svega objašnjava većim stepenom prilagođavanja INSERT naredbe SQL-a sa naredbom unosa u odgovarajuću NoSQL bazu. INSERT naredba nije adekvatno podržana ni za relacioni sistem, a slična situacija je i kod NoSQL sistema. Ovo daje mogućnosti za dalje poboljšanje sloja virtuelizacije. Takođe, ovim testom je potvrđeno da je moguće povezivanje još jednog relacionog sistema sa UnityJDBC uz, u nekim slučajevima, prihvatljivu "degradaciju" performansi.

5. ZAKLJUČAK I PLANOVI ZA BUDUĆE ISTRAŽIVANJE

Kao što je prikazano u prethodnom poglavlju eksperimentno testiranje je dovelo do nekoliko bitnih zaključaka i dalo odgovore na tri glavna pitanja koja su navedena u uvodu ovog rada.

Odgovor na prvo pitanje je potvrdan: moguće je, kao izvor, dodati novi sistema za upravljanje bazama podataka i moguće ke izvršiti integraciju korišćenjem zajedničkog sloja virtuelizacije, a konkretno implementiranog upotrebom UnityJDBC-a.

Drugo pitanje se tiče pitanja utricaja na performanse i kvantitativnim testovima koji su sprovedeni nad setovima podataka od 1.000, 10.000 i 100.000 n-torki pokazano je da uticaj na performanse postoji, da je očekivano na strani direktnog pristupa samom sloju podataka umesto putem sloja virtuelizacije, ali istovremeno i da se procenat degradacije performansi može smatrati prihvatljivim, u slučaju selektovanja velikog broja podataka, u testu za 100.000 n-torki. Degradacija performansi je još uočljivija u slučaju rada sa relativno malom količinom podataka, u testu je to bilo 1.000 n-torki. Takođe, pokazano je da se upotrebom sloja virtuelizacije pristup relacionom sistemu odvija brže nego pristup NoSQL sistemu (što je pokazalo istraživanje u referentnom radu [5]).

Odgovori na prethodna dva pitanja sugerišu i prirodan smer nastavka daljeg istraživanja. Naime, uočeno je da je Oracle relacionu bazu moguće lako i efikasno povezati sa korišćenom arhitekturom, pa se kao jedan od narednih izazova nameće testiranje mogućnosti povezivanja Oracle NoSQL sistema, posebno što Oracle NoSQL sistem za upravljanje bazom podataka nije toliko komercijalno zastupljen poput MongoDB-a, koji je uziman kao predstavnik NoSQL sistema (u referentnom, ali i u drugim radovima koji su navedeni u pregledu radova iz oblasti). Trebalo bi testirati mogućnost povezivanja takvog jednog sistema i proveriti uticaj na njegove performanse. S' obzirom da performanse NoSQL sistema evidentno predstavljaju usko grlo u referentnoj arhitekturi, bilo bi dobro razmotriti korišćenje nekih naprednijih opcija optimizacije SQL upita i prevođenja istih u NoSQL upitne jezike. Mogla bi se testirati mogućnost eksplicitnog uvođenja INESRT BULK kolekcije za baferovanje INSERT naredbi, koji bi mogao imati pozitivan uticaj na operaciju INSERT-a, koja po spomenutim rezultatima iz referentnog rada ima degradaciju od 30% kada je u pitanju korišćenje sloja virtuelizacije pri unosu podataka u NoSQL sistem. Takođe, uvođenje podrške za proceduralna proširenja SQL-a, poput PL/SQL-a za Oracle, ili T-SQL-a za Microsoft rešili bi problem nekonkurentnog unosa podataka kroz UnityJDBC u odnosu na relacioni sistem. Optimizacija upita bi se mogla poboljšati eksplicitnim operatorima npr. any, foreach i slično. Apsekt zaštite bi takođe mogao biti uzet u obzir, testirati mogućnost primene Virtual Private Database-a, koji je dostupan u Oracle-u, a koji omogućava automatsko filtriranje upita u zavisnosti od naloga sa kojim je korisnik trenutno ulogovan. Naravno, ovo su samo neki od pravaca, a svakako bi primenu spomenutih rešenja trebalo testirati na uticaj performansi i uporediti sa već ostvarenim rezultatima.

REFERENCE

[1] Codd, E: *A Relational Model of Data for Large Shared Banks*. Communications of the ACM, 13 (6), pp. 377 – 387, (1970)

[2] Strauch, C: *Nosql databases*. Stuttgart Media University, (2011)

[3] Perdue, T: *NoSQL: An Overview of NoSQL Databases*, http://newtech.about.com/od/databasemanagement/a/Nosql.htm, (pristupano: decembar 2015.)

[4] Whitepaper: *NoSQL in the Enterprise: A guide for Technology Leaders and Decision-Makers*. Datastax corporation, (2013)

[5] Lawrence, R: *Integration and Virtualization of Relational SQL and NoSQL Systems including MySQL and MongoDB*. International Conference on Computational Science and Computational Intelligence, pp.285-290, (2014)

[6] Potey, M; Digrase, M; Deshmukh, G; Nerkar, M: *Database Migration from Structured Database to non-Structured Database*. IJCA Proceedings on International Conference on Recent Trends & Advancements in Engineering Technology (ICRTAET 2015)

[7] Alomari, E; Barnawi, A; Sakr, S: *CDPort: A Portability Framework for NoSQL Datastores*. Computer Engineering And Computer Science Arabian Journal for Science and Engineering, 40 (9), pp. 2531-2553, (2015)

[8] Wu, C. M; Huang, Y. F; Lee, J: *Comparisons Between MongoDB and MS-SQL Databases on the TWC Website*. American Journal of Software Engineering and Applications, 4 (2), pp. 35 – 41, (2015)

[9] Roijackers, J; Fletcher, G.H.L: *On bridging relational and document-centric data stores*. Poglavlje u: Gottlob, G, Grasso, G, Olteanu, D, Schallhart, C (eds.) Big Data, Number 7968 in Lecture Notes in Computer Science, pp. 135–148. Springer, Berlin (2013)

[10] Livenson, I, Laure, E: *Towards transparent integration of heterogeneous cloud storage platforms*. Proceedings of the Fourth International Workshop on Data-intensive Distributed Computing (DIDC 2011), pp. 27–34, ACM, New York, NY (2011)

[11] Loutas, N; Kamateri, E; Tarabanis, K: *A semantic interoperability framework for cloud platform as a service*. IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom), pp. 280–287 (2011)

[12] Nyati, S.S; Pawar, S; Ingle, R: *Performance Evaluation of Unstructured NoSQL data over distributed framework*. International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 1623-1627, (2014)
[13] Xiaolin, W; Haopeng, C; Zhenhua, Z: *Research on Improvement of Dynamic Load Balancing in MongoDB*. IEEE 11th International Conference on Dependable, Autonomic and Secure Computing (DASC), pp.124-130, (2013)
[14] Brewer, E: *Towards Robust Distributed Systems*. Keynote at the ACM Symposium on Principles of Distributed Computing (PODC), Portland, Oregon, (2000)
[15] Gilbert, S; Lynch, N: *Perspectives on the CAP Theorem*. Computer, 45 (2), pp. 30-36, (2012)
[16] Bondi, A: *Characteristics of scalability and their impact on performance*. Proceedings of the 2nd international workshop on Software and performance, pp. 195 – 203, (2000)
[17] Hoff, T: *A Yes for a NoSQL Taxonomy* http://highscalability.com/blog/2009/11/5/a-yes-for-a-nosql-taxonomy.html, (pristupano: januar 2016.)
[18] Dukić, N: *Visoka dostupnost informacionog sistema*. INFO M, Časopis za informacione tehnologije i multimedijalne sisteme, Beograd, Br. 55/2015 (2015)
[19] Trinić, N; Stričević L; Hajduković, M: *Problem integracije visoko dostupnih kompleksnih distribuiranih softverskih sistema*. INFO M, Časopis za informacione tehnologije i multimedijalne sisteme, Beograd, Br. 45/2013 (2013)
[20] Winder, D: *Securing NoSQL applications: Best practises for big data security*, Iz: Guide to NoSQL databases: How they can help users meet big data needs http://www.computerweekly.com/tip/Securing-NoSQL-applications-Best-practises-for-big-data-security (pristupano: februar 2016.)
[21] Korać, D: *Komparacija modela zaštite informacija*. INFO M, Časopis za informacione tehnologije i multimedijalne sisteme, Beograd, Br. 56/2015 (2015)
[22] Whitepaper: *Current Data Security Issues of NoSQL Databases*. Fidelis Cybersecurity, (2014)

Srđa Bjeladinović, asistent, Fakultet organizacionih nauka Univerziteta u Beogradu.
Kontakt: srdja.bjeladinovic@fon.bg.ac.rs
Oblasti interesovanja: Relacione i NoSQL baze podataka, Metodologije razvoja informacionih sistema, Modelovanje poslovnih sistema, Integrisana softverska rešenja

Zoran Marjanović, redovni profesor, Fakultet organizacionih nauka Univerziteta u Beogradu.
Kontakt: zoran.marjanovic@fon.bg.ac.rs
Oblasti interesovanja: Baze podataka, Metodologije razvoja IS, Interoperabilnost

UPUTSTVO ZA PRIPREMU RADA

1. Tekst pripremiti kao Word dokument, A4, u kodnom rasporedu 1250 latinica ili 1251 ćirilica, na srpskom jeziku, bez slika. Preporučen obim – oko 10 strana, single prored, font 11.
2. Naslov, abstakt (100-250 reči) i ključne reči (3-10) dati na srpskom i engleskom jeziku.
3. Jedino formatiranje teksta je normal, bold, italic i bolditalic, VELIKA i mala slova (tekst se naknadno prelama).
4. Mesta gde treba ubaciti slike, naglasiti u tekstu (Slika1...)
5. Slike pripremiti odvojeno, VAN teksta, imenovati ih kao u tekstu, radi identifikacije, u sledećim formatima: rasterske slike: jpg, tif, psd, u rezoluciji 300 dpi 1:1 (fotografije, ekranski prikazi i sl.), vektorske slike – cdr, ai, fh,eps (šeme i grafikoni).
6. Autor(i) treba da obavezno priloži svoju fotografiju (jpg oko 50 Kb), navede instituciju u kojoj radi, kontakt i 2-4 oblasti kojima se bavi.
7. Maksimalni broj autora po jednom radu je 5.

Redakcija časopisa Info M