PODATKOVNE BAZE VRSTE TRIPLESTORE

Aljaž Zrnec, Dejan Lavbič, Neli Blagus, Marko Janković, Marko Bajec
Univerza v Ljubljani , Fakulteta za računalništvo in informatiko,
Laboratorij za podatkovne tehnologije, Tržaška 25, 1000 Ljubljana
aljaz.zrnec, dejan.lavbič, neli.blagus, marko.jankovic, marko.bajec@fri.uni-lj.si

Povzetek

Triplestore podatkovne baze so namenjene shranjevanju in strežbi velike količine podatkov veliki množici uporabnikov. So skalabilne in običajno namenjene spletnim aplikacijam, kot so na primer socialna omrežja. Skalabilnost, prilagodljivost in visoka razpoložljivost podatkov imata v teh sistemih prednost pred konsistentnostjo. Omogočajo prilagodljivo podatkovno shemo, zato so primerne za aplikacije s pogostimi spremembami v podatkovnem modelu. Toga shema relacijskih podatkovnih baz bi nas v takih aplikacijah ovirala. Prinašajo številne prednosti, vendar niso univerzalna rešitev. V nekaterih sistemih, kot so bančni ali borzni, so relacije in ACID transakcije še vedno potrebne.

V prispevku smo predstavimo razlike med triplestore in relacijskimi podatkovnimi bazami. V nadaljevanju se osredotočimo na primerjavo lastnosti sedmih triplestore podatkovnih baz in na primerjavo njihovih zmogljivosti. Na podlagi ugotovitev podamo priporočila, kdaj je smiselno uporabiti katero izmed rešitev.

Abstract

TRIPLE STORES

Triple store databases are intended for storing and serving large amounts of data to a large set of users. They are scalable and usually designed for web applications, such as social networks. Scalability, flexibility, and high availability are more important than data consistency in these systems. Triple stores have flexible data schema, so they are suitable for applications with frequent changes in the data model. The rigid relational schema would present an obstacle in such applications. Triple stores have many advantages over relational databases but they are not a universal solution. In some systems, such as a bank or broker, ACID transactions and relations are still needed.

In this paper we present the differences between triple stores and relational databases. In the following we focus on features of seven triple stores and we compare their performance. Based on the findings we make recommendations when it makes sense to use any of the solutions.

Ključne besede

NoSQL, relacijska podatkovna baza, triplestore podatkovna baza, semantični splet, analiza zmogljivosti, ogrodje RDF

Key words

NoSQL, relational database, triple store database, semantic web, performance analysis, RDF framework

1. UVOD

Relacijske podatkovne baze že več kot 30 let uspešno dominirajo med tehnologijami za upravljanje s podatki. Izkazale so se kot dobra rešitev za zajem in obvladovanje strukturiranih

podatkov. Njihove pomanjkljivosti se kažejo predvsem v zadnjih nekaj letih, ko smo priča nastajanju ogromnih količin nestrukturiranih podatkov. Poslovni sistemi se vedno bolj zavedajo vrednosti teh podatkov, saj jih je mogoče uporabiti za napovedovanje trendov, prepoznavanje ključnih vplivov v in izven podjetja, doseganje konkurenčne prednosti itd.

Zaradi potreb po zajemu nestrukturiranih podatkov in izvajanju analiz nad njimi, so se na področju obvladovanja podatkov pojavile podatkovne baze NoSQL. Razvoj teh baz se je v grobem razdelil na dve veji, od katerih predstavlja prva sisteme za obvladovanje velikih količin podatkov (ang. big data systems), kot sta na primer Hadoop in Hbase, druga pa sisteme semantičnega spleta, kot je na primer podatkovna shramba za shranjevanje trojčkov (ang. triplestore) [7,6]. Namen NoSQL rešitev ni nadomestiti obstoječe relacijske podatkovne baze, temveč dopolniti njihove funkcionalnosti, saj omogočajo obvladovati velike količine strukturiranih in nestrukturiranih podatkov ter izvajati kompleksne analize nad njimi. Tako relacijske kot NoSQL podatkovne baze imajo namreč svoje prednosti in slabosti, zato so njihova področja uporabe različna.

Na začetku prispevka predstavimo probleme klasičnih relacijskih podatkovnih baz. Nato sledi predstavitev novih tehnologij za obladovanje podatkov - NoSQL. V okviru tega se osredotočimo na posebno skupino NoSQL rešitev - triplestore podatkovne baze [4,1]. V nadaljevanju predstavimo lastnosti triplestore podatkovnih baz in jih med seboj primerjamo po lastnostih ter zmogljivostih. V zaključku prispevka podamo ključne ugotovitve in priporočila glede uporabe.

2. RELACIJSKA PODATKOVNA BAZA

Relacijska podatkovna baza predstavlja zbirko strukturiranih podatkov, do katerih se dostopa z uporabo standardiziranega poizvedovalnega jezika SQL, ki je z vidika uporabe relativno enostaven in široko sprejet. Relacijske podatkovne baze so še do nedavnega predstavljale ključno tehnologijo za obvladovanje strukturiranih podatkov v poslovnih sistemih, zaradi zagotavljanja pravilnosti podatkov, razpoložljivosti in transakcij, ki zadoščajo lastnostim atomarnosti, konsistentnosti, izolacije in trajnosti (angl. Atomicity, Consistency, Isolation, Durability - ACID). Slabše ali pa celo za neuporabne pa se izkažejo v primerih zahtev po visoki skalabilnosti in strežbi velike množice zahtev milijonom uporabnikov.

3. TRIPLESTORE PODATKOVNA BAZA

Triplestore predstavlja splošno ime za sisteme, ki so namenjeni shranjevanju podatkov v obliki trojčkov. Triplestore podatkovna baza je optimizirana za shranjevanje in poizvedovanje po trojčkih v standardizirani obliki RDF (ang. Resource Description Framework). Trojček predstavlja enoto RDF podatkov, ki je sestavljena iz treh komponent, subjekta, predikata in objekta (ang. subject-predicate-object), na primer "Miha lives in Ljubljana", kjer "Miha" predstavlja subjekt, "lives in" predikat in "Ljubljana" objekt.

Ogrodje RDF

Ogrodje RDF predstavlja formalizem za predstavitev poljubnih strukturiranih in ne strukturiranih podatkov o čemerkoli, kar je mogoče identificirati z uporabo enotnih označevalnikov virov - URI (ang. Uniform Resource Identifier). Podatkovni model ogrodja RDF temelji na grafih. RDF predstavi vozlišča v grafu z uporabo enotnih označevalnikov virov (v nadaljevanju URI). Vsaka stvar, ki je opisana z uporabo RDF, mora imeti svoj URI.

Prav tako morajo imeti atributi URI-ja (predikati) lasten URI. Vrednost atributa (objekt) je lahko URI ali literal, na primer niz ali število. Ker trojčki vsebujejo predikate, je nad njimi mogoče uporabiti pravila sklepanja. Triplestore podatkovne baze predstavljajo idealno tehnologijo za semantični splet, zaradi česar se jih je prijel tudi naziv semantične podatkovne baze.

Arhitektura

Z vidika arhitekture je mogoče triplestore podatkovne baze razdeliti na izvirne RDF shrambe, hibridne RDF shrambe s podporo relacijskih podatkovnih baz in RDF ovojnice.

Izvirne RDF shrambe so namenjene procesiranju RDF podatkov. Z vidika zgradbe so izdelane povsem na novo in delujejo neodvisno od drugih SUPB. Njihov namen je shranjevanje in strežba velike količine podatkov velikemu številu uporabnikov. So skalabilne in običajno namenjene spletnim aplikacijam, na primer socialna omrežja. Skalabilnost in visoka razpoložljivost imata v teh shrambah prednost pred konsistentnostjo podatkov. Omogočajo prilagodljivo podatkovno shemo, zato so primerne za aplikacije s pogostimi spremembami v podatkovnem modelu. Toga shema relacijskih podatkovnih baz bi nas v takih aplikacijah ovirala [9]. Prinašajo številne prednosti, vendar so v nekaterih sistemih, kot so bančni ali borzni, relacije in ACID transakcije še vedno potrebne. Podatki morajo biti vedno razpoložljivi in pravilni [8], izvirne RDF shrambe pa s sprejetjem možne začasne nekonsistentnosti pridobijo pri skalabilnosti in prilagodljivosti.

Hibridne RDF shrambe uporabljajo za shranjevanje in dostop do podatkov funkcionalnosti relacijskega SUPB. Delimo jih na sisteme z generično ali ontološko specifično shemo. Sisteme z generično shemo je nadaljnje mogoče razdeliti na model z eno tabelo, model s tabelo lastnosti, normaliziran model in hibridni pristop. Ontološko-specifične sheme ne shranjujejo trojčkov v enotno tabelo, temveč uporabljajo pri shranjevanju shemo, ki posnema strukturne lastnosti ontologije [5]. Vsaka sprememba ontologije se prenese na strukturo podatkovnih tabel.

RDF ovojnice so posebne programske komponente, ki delujejo nad obstoječim izvorom podatkov. Omogočajo dostop do podatkov v obliki RDF, ne da bi se izvorni podatki morali spreminjati. Slabost te rešitve je, da so podatki namenjeni samo branju.

Prilagodljivost

Relacijske podatkovne baze so zaradi stroge strukturiranosti podatkov zelo neprilagodljive. Če hočemo med podatki ustvariti nove relacije, je potrebno prenoviti shemo in dodati povezovalne tabele. Triplestore omogoča dodati novo relacijo z dodajanjem novega trojčka, pri čemer ni potrebno spreminjati sheme, niti dodati novih tabel. Slabost takega pristopa je, da bi morali dodati zelo veliko trojčkov, če bi hoteli opisati vse nove relacije. Zaradi tega je hitreje, če se v triplestore doda pravila sklepanja.

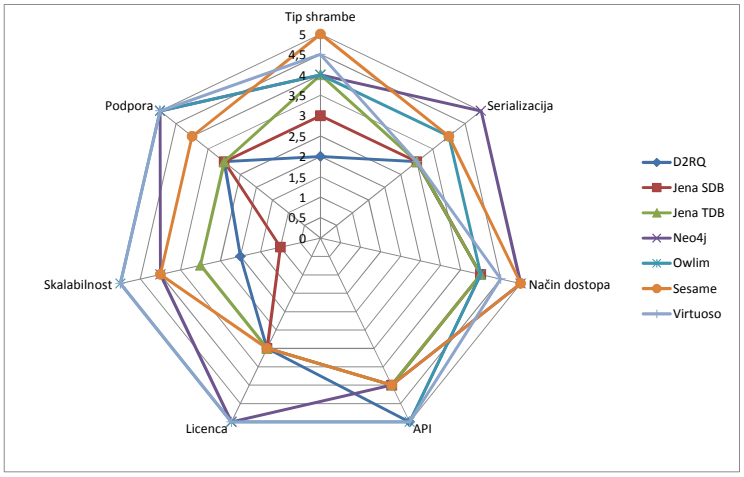
Prilagodljivost triplestore podatkovne baze se kaže tudi v tem, da je dodajanje novih podatkov, ki niso pričakovani pri prvotni zasnovi zbirke, veliko bolj enostavno. Načrtovalcem ni potrebno predhodno opredeliti sheme podatkov, ki se bodo nahajali v podatkovni bazi, ampak zgradijo ontologijo, ki temelji na podatkih v zbirki. Ontologijo je mogoče kasneje spreminjati, lahko pa se izdela tudi že na začetku.

4. PRIMERJAVA TRIPLESTORE PODATKOVNIH BAZ

V okviru primerjave triplestore podatkovnih baz smo izvedli primerjavo lastnosti in analizo zmogljivosti. Med seboj smo primerjali 7 triplestore rešitev, ki se v zadnjem času najbolj uporabljajo: D2RQ, Jena SDB, Jena TDB, Neo4j, Owlim, Sesame in Virtuoso.

Primerjava lastnosti

V okviru primerjave lastnosti triplestore podatkovnih baz smo ocenjevali naslednje lastnosti: tip shrambe, serializacija, način dostopa, API, licenca, skalabilnost in podpora. Lastnosti podatkovnih baz smo ocenili z lestvico od 0 do 5, pri čemer je 0 najslabša, 5 pa najboljša ocena. Pri oceni smo upoštevali prednosti vsake izmed podatkovnih baz pred drugimi. Primerjavo lastnosti z našimi ocenami najlepše prikazuje radarski graf na sliki Slika 1:



Slika 1: Primerjava lastnosti triplestore podatkovnih baz [6]

Analiza zmogljivosti

V okviru analize zmogljivosti smo si zamislili problemsko domeno - aplikacijo za prodajo izdelkov, ki uporablja triplestore za hranjenje podatkov o artiklih. Glede na uporabo aplikacije smo opredelili 3 testne scenarije, ki jih prikazuje Tabela 1:

Scenarij 1	Scenarij 2	Scenarij 3
Podatkovna baza se uporablja samo za branje: • branj 100%, • pisanj 0%, • brisanj 0%. Merili smo celoten čas izvajanja in čas izvedbe posamezne operacije branja.	Podatkovna baza se uporablja za uravnoteženo branje in vnašanje podatkov: • branj 50%, • pisanj 50%, • brisanj 0%. Merili smo celoten čas izvajanja, čas izvajanja samo branj/pisanj in povprečen čas izvedbe posamezne operacije branja/pisanja.	Podatkovna baza se večinoma uporablja za vstavljanje: • branj 0%, • pisanj 80%, • brisanj 20%. Merili smo celoten čas izvajanja, čas izvajanja samo pisanj/brisanj in povprečen čas izvedbe posamezne operacije pisanja/brisanja.

Tabela 1: Testni scenariji

5. TESTNO OKOLJE

Testiranje smo izvedli na operacijskem sistemu Ubuntu Linux 32 bit, različice 12.10, ki smo ga poganjali v navideznem stroju VmWare s 3 Gb pomnilnika. Vsako triplestore podatkovno

bazo smo namestili v svoj navidezni stroj, ki se je nahajal na gostitelju s procesorjem Intel Core 2 Duo 2 GHz in s 4 Gb pomnilnika.

Podatke za testiranje smo pridobili s pomočjo podatkovnega generatorja Berlin SPARQL Benchmark Data Generator [3], ki omogoča kreiranje poljubne količine podatkov v različnih RDF formatih in SQL skriptah. Testirali smo z naslednjimi količinami podatkov: 10.000, 100.000, 1.000.000 in 5.000.000 trojčkov. Merili smo čas izvedbe začetnega vstavljanja podatkov v posamezno triplestore podatkovno bazo ter hitrosti izvajanja posamičnih scenarijev.

Vsak scenarij je obsegal 1000 operacij, ki so se glede na scenarij, delile na branje, pisanje in brisanje. Za operacijo branja smo izdelali pet različnih SPARQL SELECT poizvedb in rezultat omejili na 10 trojčkov. V vsakem scenariju, ki vključuje branje, so se poizvedbe večkrat ponovile v naključnem zaporedju. Vse spremembe podatkov so potekale preko operacij iz SPARQL INSERT/UPDATE 1.1. Podatke smo preko SPARQL vstavljali z operacijo INSERT, brisali pa preko operacije DELETE. Izjema je bila podatkovna baza Neo4j, ki ne podpira vstavljanja ali brisanja podatkov preko SPARQL, zato smo pri njej za spremembe podatkov uporabili neposredne operacije nad grafom. Vsaka operacija je obsegala vnos ali brisanje enega trojčka.

Testiranje smo izvedli za vsako količino podatkov posebej. Posamezni scenarij smo izvedli petkrat. Izmed izmerjenih časov smo izločili najslabšega in najboljšega, iz preostalih treh pa smo izračunali povprečje, ki je predstavljalo čas potreben za izvedbo scenarija.

6. ANALIZA

Vstavljanje podatkov

Začetno vstavljanje podatkov je daleč najhitreje opravil D2RQ preko MySQL podatkovne baze. Pri majhnih količinah podatkov je Jena TDB lahko še sledila hitrosti D2RQ, z večanjem količine podatkov pa se je razlika v porabljenem času vstavljanja povečevala od 2-krat pri 1.000.000 do skoraj 4-krat pri 5.000.000 trojčkih (Tabela 2). Jena SDB tako kot D2RQ uporablja MySQL podatkovno bazo, vendar je vstavljanje trojčkov skoraj 100-krat počasnejše pri Jena SDB. Iz tega je mogoče sklepati, da je za shranjevanje RDF podatkov primernejša namenska podatkovna baza. Za majhne količine podatkov, to je do 100.000 vnosov, so Neo4j, Owlim in Sesame dosegli primerljive rezultate. Pri večjih količinah podatkov je Neo4j začel znatno zaostajati in dosegel nekajkrat daljše čase vstavljanja. Iz podatkov lahko sklepamo, da bi se z večanjem količine podatkov razlika v času vstavljanja še stopnjevala. Owlim in Sesame, ki oba uporabljata Sesame OpenRDF API, sta dosegala približno enake rezultate do 1.000.000 trojčkov, nato je prevladala hitrost Owlim. Vnos v Virtuosovo bazo je bil že pri majhnih količinah podatkov daleč najpočasnejši, pri 5.000.000 podatkov pa vnos sploh ni bil uspešno dokončan, saj je pri vnosu porabil celoten razpoložljiv pomnilnik.

Št. trojčkov	D2RQ	JenaSDB	JenaTDB	Neo4j	Owlim	Sesame	Virtuoso
10.000	1,100	3,673	1,045	13,239	6,920	8,358	100,054
100.000	2,800	44,597	4,998	58,480	63,302	59,370	963,969
1.000.000	23,500	1359,185	44,148	2222,800	578,833	680,320	9568,958
5.000.000	153,000	17319,072	595,167	34037,544	2969,284	4300,746	/

Tabela 2: Čas vstavljanja podatkov v sekundah

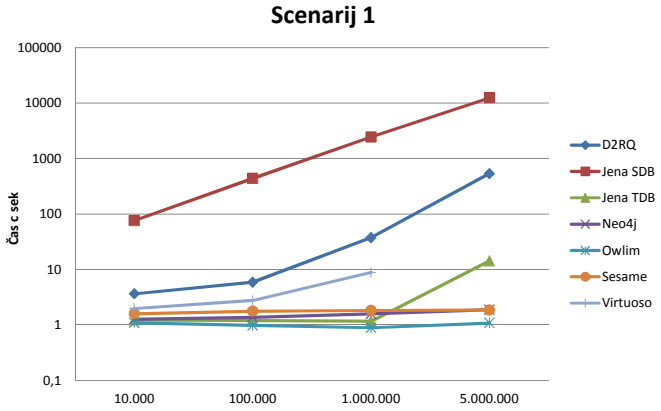
Scenarij 1

Prvi scenarij je obsegal samo branja. Merili smo celoten čas izvajanja 1000 branj in povprečen čas izvedbe posameznega branja. D2RQ in Jena SDB, ki za shranjevanje podatkov uporabljata MySQL podatkovno bazo, sta za izvajanje poizvedb porabila precej daljši čas od ostalih rešitev (Slika 2). Skupni čas izvajanja se je pri teh dveh podatkovnih bazah najhitreje povečeval z naraščanjem količine podatkov (Tabela 3). Owlim je dosegel najboljše rezultate pri vseh testiranih količinah podatkov. Skupni čas izvajanja poizvedb preko Owlim-a se z naraščanjem količine podatkov bistveno ni podaljševal in je ostal okoli sekunde. Najslabše rezultate od rešitev z izvirnimi repozitoriji je dosegel Virtuoso, in sicer skorajda 10-krat počasnejši čas izvajanja od najhitrejšega Owlima že pri 1.000.000 podatkov. Jena SDB je s povprečnim časom branja več kot 12 sekund pri 5.000.000 vnosih daleč najpočasnejša in neprimerna za uporabo v primeru naše problemske domene. Owlim, Sesame, Neo4j in Jena TDB z zelo nizkimi povprečnimi časi na poizvedbo zadostujejo tudi potrebam naše problemske domene.

V testu Berlin SPARQL Benchmark V2 so testirali Jena TDB, Sesame, Virtuoso in D2RQ podatkovne baze. Tako kot pri našem testiranju je med njimi najpočasneje s scenarijem opravil D2RQ, Sesame pa je bil najhitrejši. V nasprotju z našimi rezultati, kjer je bil Virtuoso počasnejši od Jena TDB, je pri njihovem testu Virtuoso izvedel poizvedbe hitreje [2].

Število trojčkov		D2RQ	Jena SDB	Jena TDB	Neo4j	Owlim	Sesame	Virtuoso
10.000	Čas izvajanja	3,640	76,322	1,229	1,274	1,095	1,594	1,980
	Povpr. čas br.	0,004	0,076	0,001	0,001	0,001	0,002	0,002
100.000	Čas izvajanja	5,895	436,902	1,198	1,371	0,984	1,779	2,766
	Povpr. čas br.	0,006	0,437	0,001	0,001	0,001	0,002	0,003
1.000.000	Čas izvajanja	37,503	2435,792	1,162	1,583	0,889	1,829	8,842
	Povpr. čas br.	0,038	2,436	0,001	0,002	0,001	0,002	0,009
5.000.000	Čas izvajanja	536,094	12390,580	14,278	1,887	1,085	1,864	/
	Povpr. čas br.	0,536	12,3906	0,014	0,002	0,001	0,002	/

Tabela 3: Časi izvedbe scenarija 1 za različne množice trojčkov v sekundah



Slika 2: Čas izvajanja scenarija 1

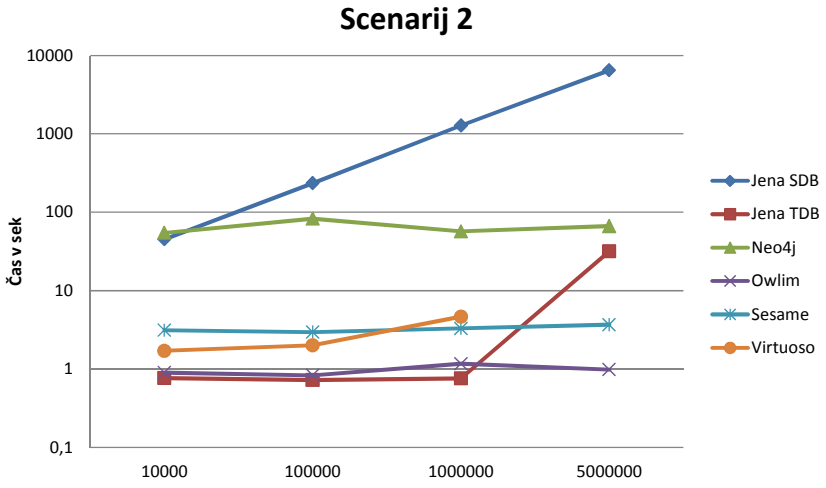
Scenarij 2

V okviru drugega scenarija smo izvedli 500 branj in 500 vstavljanj podatkov. Merili smo celoten čas izvajanja, čas izvajanja samo branj/pisanj in povprečen čas izvedbe posamezne operacije branja/pisanja.

Najboljše rezultate sta pri 10.000 trojčkih dosegla Jena TDB in Owlim, zaradi hitrejšega branja je boljši rezultat dosegla Jena TDB (Tabela 4). Najpočasnejša sta bila Neo4j in Jena SDB. Neo4j zaradi izredno počasnega časa vstavljanja, ki je več kot 10-krat daljši kot pri ostalih, Jena SDB pa zaradi počasnega branja. Pri 100.000 in 1.000.000 trojčkov sta Jena TDB in Owlim dosegla približno enake skupne čase izvajanja, približno dve sekundi počasneje je s scenarijem zaključil Sesame (Slika 3). S povečevanjem količine podatkov se je čas branja v Jena SDB hitro povečeval in pri 5.000.000 podatkov je bil povprečni čas enega branja skorajda enak celotnemu času vstavljanja. Čas izvajanja scenarija v Jena TDB se je glede na 1.000.000 podatkov pri 5.000.000 podaljšal skoraj 40-krat. Vzrok je predvsem v počasnejšem branju, ki je bilo pri Jena TDB približno 3-krat počasnejše od vstavljanja. Sesame je kot drugi najhitrejši pri 5.000.000 podatkov dosegel približno 4-krat počasnejši rezultat kot Owlim. Vzrok je v 4-krat počasnejšem vstavljanju v primerjavi z branjem. Owlim je imel najboljše razmerje med hitrostjo branja in vstavljanja in je s scenarijem opravil najhitreje.

Število trojčkov		Jena SDB	Jena TDB	Neo4j	Owlim	Sesame	Virtuoso
10.000	Čas izvajanja	45,095	0,771	54,660	0,900	3,132	1,712
	Čas branja	38,396	0,433	0,964	0,567	0,658	0,780
	Čas vstavlj.	6,699	0,337	53,696	0,333	2,475	0,932
	Povpr. čas br.	0,077	0,001	0,002	0,001	0,001	0,002
	Povpr. čas vst.	0,013	0,001	0,107	0,001	0,005	0,002
100.000	Čas izvajanja	235,267	0,727	83,060	0,832	2,960	2,012
	Čas branja	228,882	0,418	0,490	0,433	0,671	1,083
	Čas vstavlj.	6,385	0,309	82,570	0,398	2,289	0,928
	Povpr. čas br.	0,458	0,001	0,001	0,001	0,001	0,002
	Povpr. čas vst.	0,013	0,001	0,165	0,001	0,005	0,002
1.000.000	Čas izvajanja	1277,367	0,760	57,076	1,173	3,297	4,663
	Čas branja	1270,262	0,442	0,616	0,538	0,780	3,730
	Čas vstavlj.	7,105	0,319	56,460	0,635	2,518	0,932
	Povpr. čas br.	2,541	0,001	0,001	0,001	0,002	0,007
	Povpr. čas vst.	0,014	0,001	0,113	0,001	0,005	0,002
5.000.000	Čas izvajanja	6456,452	31,784	66,763	0,985	3,687	/
	Čas branja	6443,575	22,317	0,701	0,456	0,709	/
	Čas vstavlj.	12,876	9,467	66,062	0,528	2,978	/
	Povpr. čas br.	12,887	0,045	0,001	0,001	0,001	/
	Povpr. čas vst.	0,026	0,019	0,132	0,001	0,006	/

Tabela 4: Časi izvedbe scenarija 2 za različne množice trojčkov v sekundah



Slika 3: Čas izvajanja scenarija 2

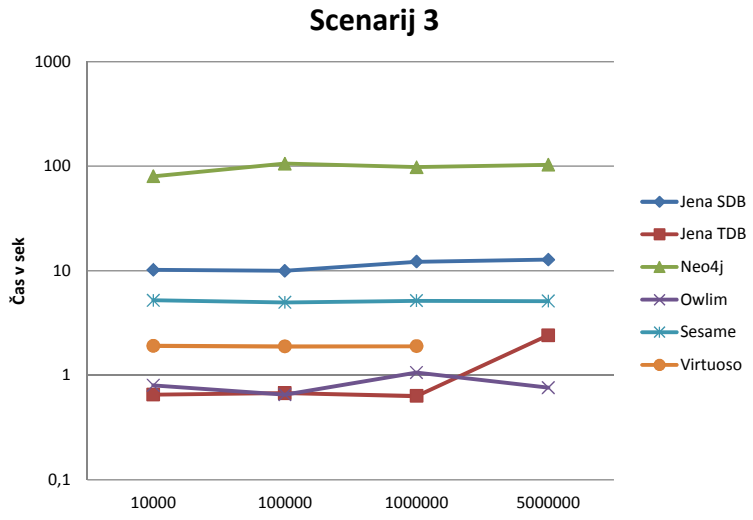
Scenarij 3

V okviru tretjega scenarija smo izvedli 800 pisanja in 200 operacij brisanja podatkov. Merili smo celoten čas izvajanja, čas izvajanja samo pisanj/brisanj in povprečen čas izvedbe posamezne operacije pisanja/brisanja.

Najhitreje sta scenarij izvedla Jena TDB in Owlim (Slika 4Slika 3). Jena TDB je bila nekoliko hitrejša do količine 1.000.000 podatkov, s krajšim časom vstavljanja podatkov. Najpočasneje je scenarij izvedel Neo4j zaradi počasnega vstavljanja podatkov (Tabela 5). Pri 100.000 podatkih je bil Owlim nekoliko boljši od Jena TDB, pri 5.000.000 podatkov pa je bil Owlim približno 3-krat hitrejši od Jena TDB. Sesame je preko celega scenarija zaostajal za najhitrejšima, zaradi nekajkrat počasnejšega vstavljanja podatkov. Povprečni čas izvedbe ene operacije brisanja je zelo majhen pri vseh rešitvah, nekoliko večji je čas vstavljanja pri Neo4j. Sesame in Virtuoso sta skozi celoten scenarij dosegala konstanten čas izvajanja.

Število trojčkov		Jena SDB	Jena TDB	Neo4j	Owlim	Sesame	Virtuoso
10.000	Čas izvajanja	10,196	0,654	79,818	0,803	5,203	1,910
	Čas vstavljanja	8,823	0,510	78,841	0,680	3,550	1,437
	Čas brisanja	1,372	0,143	0,977	0,122	1,653	0,437
	Povpr. čas vst.	0,011	0,001	0,099	0,001	0,004	0,002
	Povpr. čas bris.	0,007	0,001	0,005	0,001	0,008	0,002
100.000	Čas izvajanja	9,954	0,676	105,452	0,653	4,976	1,890
	Čas vstavljanja	8,526	0,506	102,624	0,521	3,417	1,422
	Čas brisanja	1,428	0,170	2,828	0,132	1,559	0,469
	Povpr. čas vst.	0,011	0,001	0,128	0,001	0,004	0,002
	Povpr. čas bris.	0,007	0,001	0,014	0,001	0,008	0,002
1.000.000	Čas izvajanja	12,200	0,634	97,724	1,062	5,169	1,896
	Čas vstavljanja	10,267	0,457	94,872	0,707	3,557	1,421
	Čas brisanja	1,933	0,176	2,852	0,355	1,612	0,474
	Povpr. čas vst.	0,013	0,001	0,119	0,001	0,004	0,002
	Povpr. čas bris.	0,010	0,001	0,014	0,002	0,008	0,002
5.000.000	Čas izvajanja	12,748	2,415	103,048	0,761	5,121	/
	Čas vstavljanja	10,948	2,229	100,075	0,497	3,569	/
	Čas brisanja	1,800	0,186	2,973	0,264	1,551	/
	Povpr. čas vst.	0,014	0,003	0,125	0,001	0,004	/
	Povpr. čas bris.	0,009	0,001	0,015	0,001	0,008	/

Tabela 5: Časi izvedbe scenarija 3 za različne množice trojčkov v sekundah



Slika 4: Čas izvajanja scenarija 3

Ugotovitve

Prvi scenarij obsega le branje podatkov in najbolj primerni za take operacije so Neo4j, Owlim in Sesame. Vsi imajo zelo nizek povprečni čas branja, le nekaj tisočink sekunde. To je za uporabnika neopazno in omogoča dobro uporabniško izkušnjo. Jena SDB in D2RQ pa sta z dolgimi povprečnimi časi branja neprimerna za uporabo v aplikacijah, kjer je odzivnost zelo pomembna.

V drugem scenariju, z enako količino branja in vstavljanja, sta se za najprimernejšo izbiro izkazala Owlim in Sesame. Oba imata nizek povprečni čas branja in vstavljanja, kar dovoljuje nemoteno delovanje sistema, ki smo si ga zamislili v okviru problemske domene. Jena SDB je z nekaj sekund dolgim časom branja neprimerna za uporabo v aplikacijah, kjer je odzivnost pomembna.

V aplikacijah, kjer je veliko spreminjanja podatkov, je glede na rezultate tretjega scenarija za uporabo najprimernejši Owlim. Konstantna časa vstavljanja in brisanja pri vseh količinah podatkov omogočata odlične odzivne čase. Za uporabo sta primerna tudi malenkost počasnejša Jena TDB in Sesame, saj je povprečen čas posamezne operacije le nekaj tisočink sekunde. Neo4j zaradi počasnega vstavljanja podatkov ne more zagotoviti zadovoljivih odzivnih časov.

7. ZAKLJUČEK

Podobno kot ostale NoSQL rešitve, predstavljajo triplestore podatkovne baze razmeroma novo področje, ki se razvija z neizmerno hitrostjo. Za večino NoSQL rešitev velja, da je njihova namestitev in uporaba bolj zapletena kot pri relacijskih podatkovnih bazah. Orodja so slabše razvita in ponujajo precej manj funkcionalnosti za razliko od ustaljenih in standardiziranih relacijskih podatkovnih baz.

V okviru prispevka smo analizirali zmogljivosti triplestore podatkovnih baz pri razmeroma majhni količini podatkov, glede na njihove zmožnosti, vendar so se že tu pojavile velike razlike v hitrosti delovanja. Ugotovili smo, da rešitve, ki uporabljajo za shranjevanje relacijske podatkovne baze, po zmogljivostih ne morejo konkurirati namenskim triplestore bazam. Med triplestore podatkovnimi bazami se je za najbolj vsestranskega izkazal Owlim, pri katerem so časi branja, vstavljanja in brisanja približno enaki.

Ker so triplestore podatkovne baze z vidika implementacije tako zelo različne, je zelo težko izbrati univerzalno rešitev, saj ima vsaka od njih svoje prednosti in slabosti. Zaradi tega je pred odločitvijo smiselno preizkusiti nekaj rešitev, ki po funkcionalnostih najbolj ustrezajo našim potrebam. Pri odločitvi si lahko pomagamo z lastnostmi, ki smo jih predstavili v okviru prispevka. Rezultati testiranja so nastali na podlagi izvajanja scenarijev na specifični strojni konfiguraciji za točno določeno problemsko domeno. Zaradi tega so primerni predvsem kot opora pri iskanju primerne implementacije podatkovne baze. Pri sprejemanju končne odločitve zato priporočamo lastno testiranje na ciljni strojni konfiguraciji.

Naj na koncu podamo še zanimivo vprašanje dr. Jans Aasmana: Recimo, da bi morali kupiti nov avtomobil. Seveda bi se odločili za enak ali podoben avtomobil kot ga že imate v garaži, ker se proizvaja že 30 let, se odlično obnaša na cesti, ima nizko porabo, popraviti pa ga znajo v vsaki mehanični delavnici. Ampak! Na trgu se je pojavil nov avtomobil s podobnimi lastnostmi, primerljivo ceno, a veliko bolj prilagodljiv, saj lahko leti in se premika pod vodo. Za katerega bi se odločili?

VIRI IN LITERATURA

[1] ALLEMANG, D., HENDLER, J.: Semantic Web for the Working Ontologist Modeling in RDF, RDFS and OWL, Elsevier Inc., 2008, str.: 31-57, ISBN-13: 978-0-12-373556-0.

[2] Berlin SPARQL Benchmark VS: Dostopno 5.1.2013 na: http://wifo5-03.informatik.uni-mannheim.de/bizer/berlinsparqlbenchmark/V2/results/index.html#results.

[3] Data Generator and Test Driver: Dostopno 5.1.2013 na: http://www4.wiwiss.fu-berlin.de/bizer/BerlinSPARQLBenchmark/spec/BenchmarkRules/index.html#datagenerator.

[4] DUCHARME, B.: Learning SPARQL, O'REILLY, str. 19-44, ISBN: 978-1-449-30659-5, 2011.

[5] HASLHOFER, B., E., MOMENI, B. , SCHANDL, S., ZANDER: European RDF Store Report, University of Vienna, Dostopno 16.1.2013 na: http://eprints.cs.univie.ac.at/2833/. 2011.

[6] PUNTAR, S.: Pregled in primerjava triplestore podatkovnih baz, Diplomsko delo, Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, 2012.

[7] W3C (2013). RDF 1.1 Concepts and Abstract Syntax, Dostopno 16.1.2013 na http://www.w3.org/TR/2013/WD-rdf11-concepts-20130115

[8] ZRNEC, A., D., LAVBIČ, L., ŠUBELJ, S., ŽITNIK, A., KUMER, M., BAJEC: Podatkovne baze NOSQL. V: 19. konferenca Dnevi slovenske informatike, Portorož, 16.-18. april 2012. Ustvarimo nove rešitve! : zbornik prispevkov. 1. izd. Ljubljana: Slovensko društvo Informatika, str. 1-10, graf. prikazi.

[9] ŽLENDER, R.: Primerjava zmogljivosti ne-relacijskih podatkovnih baz, Diplomsko delo, Fakulteta za računalništvo in informatiko. Univerza v Ljubljani, 2011.