Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004:007
DOI: https://doi.org/10.24867/06OI02Spasojevic

UPOREDNA ANALIZA UTICAJA TEHNIKA INDEKSIRANJA NA PERFORMANSE IZVRŠENJA UPITA U RAZLIČITIM SISTEMIMA ZA UPRAVLJANJE RELACIONIM BAZAMA PODATAKA

COMPARATIVE IMPACT ANALYSIS OF INDEX TECHNIQUES ON QUERY EXECUTION PERFORMANCE IN DIFFERENT RELATIONAL DATABASE MANAGEMENT SYSTEMS

Ivana Spasojević, *Fakultet tehničkih nauka, Novi Sad*

Oblast – INŽENJERSTVO INFORMACIONIH SISTEMA

Kratak sadržaj – *U radu je dat uvid u osnovnu problematiku podešavanja performansi i optimizacije upita nad bazom podataka. Objašnjene su osnovne vrste indeksnih struktura koje se mogu koristiti u procesu optimizacije upita. Dat je prikaz sistema za upravljanje bazama podataka i detaljnije objašnjeni relacioni SUBP korišćeni u ovom radu. Specificirani su slučajevi upotrebe na osnovu kojih je izvršeno testiranje performansi u tri različita SUBP-a. Na kraju, prokomentarisani su i upoređeni dobijeni rezultati.*

Ključne reči: *Baza podataka, optimizacija, indeksi, performanse.*

Abstract – *In this paper we give an insight into the basic issues of tuning performance and query optimization within a database. The basic types of index structures that can be used in the query optimization process are explained. The database management systems and the relational DBMSs used in this paper are presented. Use cases were specified to perform performance testing in three different DBMSs. Finally, the results obtained are shown and compared.*

Keywords: *Database, optimization, indexes, performances.*

1. UVOD

Informacione tehnologije i procesi digitalizacije preuzimaju dominantnu ulogu u svakodnevnom životu ljudi. Računari i mobilni telefoni postaju sve „pametniji", a njihova upotreba učestalija. Primena informacionih tehnologija u gotovo svim aspektima života i rada omogućila je da se lako i brzo generišu velike količine podataka i informacija.

Sa razvojem tehnologije rastu i naše potrebe za podacima i informacijama, a paralelno sa time i potreba za trajnim memorisanjem podataka, kako bismo mogli da ih koristimo i u budućnosti. Takođe, veoma je značajno obezbediti pravovremeni pristup podacima i informacijama, vodeći računa o pravima pristupa podacima.

Neretko se dešava da se korisnik pri upotrebi aplikacija susreće sa situacijom u kojoj mora da čeka određen vremenski period na odziv aplikacije, odnosno neophodno je da prođe određeno vreme od trenutka slanja nekog zahteva aplikaciji od strane korisnika do trenutka u kojem aplikacija daje odgovor na dati zahtev. Proces identifikacije uzroka problema pada performansi aplikacije za obradu podataka smeštenih u nekoj bazi podataka je zahtevan, a može biti vezan za: hardver, operativni sistem, aplikaciju, sistem za upravljanje bazom podataka (SUBP), projektovanu i implementiranu šemu baze podataka, kao i za način realizacije fizičke strukture baze podataka. Postoje različiti parametri performantnog rada SUBP-a i ovaj rad fokusiran je na jedan od njih – efikasnost realizacije upita nad bazom podataka. Savremeni sistemi za upravljanje bazama podataka u najvećoj meri su zasnovani na relacionom ili objektno-relacionom modelu podataka. Svi SUBP-ovi koji su prezentovani u ovom radu inicijalno su bili zasnovani na relacionom modelu podataka i koriste različite tehnike indeksiranja kako bi obezbedili efikasniju selekciju i manipulisanje sadržajem baze podataka.

Cilj ovog rada je da kroz uporednu analizu uticaja tehnika indeksiranja na performanse izvršenja upita u različitim sistemima za upravljanje relacionim bazama podataka obezbedi potrebne informacije za donošenje odluke o izboru SUBP-a i tehnike indeksiranja koja će unaprediti efikasnost upita nad bazom podataka. Kako bi se ostvario postavljeni cilj, neophodno je proučiti teorijske i praktične aspekte tehnika indeksiranja i optimizacije upita u relacionim bazama podataka uz posvećivanje posebne pažnje *Microsoft SQL Server*, *Oracle* i *PostgreSQL* SUBP-ovima. Potrebno je i specificirati slučajeve upotrebe koji će omogućiti ilustraciju primene tehnika indeksiranja u različitim sistemima za upravljanje relacionim bazama podatka kroz njihovu implementaciju u okviru tri prethodno pomenuta SUBP-a.

2. ORGANIZACIJA PODATAKA U BAZI PODATAKA

Fizička struktura baze podataka, odnosno izbor načina organizacije podataka u datotekama, može značajno da utiče na performantnost baze podataka. Organizacija datoteke predstavlja način smeštanja slogova i blokova u fizičke blokove na eksternoj memoriji i njihovog međusobnog povezivanja.

U osnovne organizacije datoteka spadaju serijska, sekvencijalna, spregnuta i rasuta organizacija datoteka. Ukoliko se neka osnovna organizacija datoteka proširi nekim pomoćnim strukturama tada nastaje složena organizacija datoteka. Primeri složenih organizacija datoteka su rasuta organizacija sa zonom prekoračenja, statička indeksna i dinamička indeksna organizacija datoteka. Kako bi se izvršila razmena podataka između eksternih memorijskih uređaja i operativne memorije,

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bila dr Sonja Ristić, red.prof.

potrebno je pristupiti podacima koji su organizovani na neki od pomenutih načina organizacije datoteka. Metode pristupa predstavljaju skup programa, koji učestvuje u izgradnji i održavanju strukture datoteke.

Postoje tri vrste metoda pristupa: sekvencijalni (redosledni) pristup, direktni pristup i dinamički (kombinovani) pristup [5]. Dinamičke indeksne organizacije datoteka su složene organizacije datoteka kod kojih je zona podataka, u koju se smeštaju slogovi datoteke, najčešće realizovana kao serijska datoteka. Zona indeksa je realizovana kao stablo traženja, a SUBP-ovi najčešće koriste balansirano stablo traženja, po pravilu B-stablo, ili neku od njegovih modifikovanih verzija. Stabla traženja služe za pretragu slogova smeštenih na eksternoj memoriji i najčešće sadrže jedinstvene vrednosti obeležja od kojih je svaka od njih pokazivačemka zoni podataka povezana sa odgovarajućim slogom. Vrednosti smeštene u čvorovima stabla su uređene tako da se u podstablu na koji ukazuje njima pridruženi pokazivač mogu naći samo vrednosti veće od posmatrane, a manje od naredne vrednosti u nizu.

Savremeni SUBP-ovi najčešće koriste B^+-stabla kao dinamičke indeksne strukture. Listovi stabla su uglavnom dvostruko spregnuti. Postoje i takve implementacije B^+-stabala kod kojih se u listovima nalaze kompletni slogovi, pa takva stabla čine istovremeno i zonu podataka.

3. PROCES OPTIMIZACIJE UPITA

Osnovni cilj sprovođenja procesa optimizacije određenog upita nad bazom podataka je smanjenje vremena odziva sistema na pristigle zahteve. Ulazne specifikacije za optimizator upita mogu, između ostalog, da sadrže sledeće komponente: tekst sintaksno korektne *SELECT* naredbe upita, opis šeme baze podataka i fizičke strukture potrebnog dela baze podataka koji su smešteni u rečniku podatka, pri čemu je posebno značajna specifikacija potencijalno korisnih indeksa poput struktura tipa B^+ stabla i brojeve torki u relevantnim tabelama za upit.

Osnovni koraci u procesu optimizacije upita su sledeći: korisnik definiše SQL upit koji se optimizuje a zatim se prelazi na korak parsiranje, u okviru kog SUBP prevodi kod napisan u SQL jeziku u internu reprezentaciju, zasnovanu na relacionoj algebri. Dalje se ovako zapisan kod validira kako bi se utvrdilo da li su ispoštovana sva sintaksna i semantička pravila za pisanje upita. Ako u ovom procesu nisu utvrđena nikakva odstupanja, prelazi se na proces optimizacije upita. Generisanje logičkog plana optimizacije upita podrazumeva analizu različitih operatora koji se javljaju u upitu, poput projekcije, spoja i selekcije. Logička optimizacija se često naziva i algebarska optimizacija i ona podrazumeva nalaženje najpovoljnijeg redosleda obavljanja algebarskih operacija primenom ekvivalentnih transformacija relacione algebre [4].

Na kraju, bira se optimalan plan izvršavanja upita. Ovaj plan se može posmatrati kao stablo čiji svaki čvor sadrži određenu operaciju iz upita koji se izvršava, a kojoj je pridružen odgovarajući operator baze podataka (npr. skeniranje upotrebom indeksa, filter ili agregacija).

4. SISTEMI ZA UPRAVLJANJE BAZAMA PODATKA (SUBP)

Sistem za upravljanje bazom podataka je zapravo program ili grupa programa čija namena je povećanje pouzdanosti i efikasnosti upotrebe baze podataka kroz njeno definisanje, kreiranje, zaštitu i održavanje [1]. SUBP treba da omogući, između ostalog, i: logičku i fizičku nezavisnost programa i podataka, konkurentno korišćenje baze podataka od strane više ovlašćenih korisnika, performantno korišćenje baze podataka i njeno distribuiranje na više servera povezanih putem računarske mreže [6]. SUBP podržava jezike za opis i održavanje šeme baze podataka DDL (*Data Definition Language*) i za manipulaciju podacima DML (*Data Manipulation Language*). Kako bi SUBP efikasno izvršavao osnovne zadatke za koje je zadužen, veoma su važne njegove karakteristike – adaptivnost, pouzdanost i performantnost. Pored toga, svaki SUBP zasnovan je na nekom modelu podataka. Kod savremenih SUBP-ova najčešće je to relacioni model podataka pa SUBP samim tim, upravlja relacionim bazama podataka u kojima su podaci smešteni u okviru međusobno povezanih dvodimenzionalnih tabela, gde svaka od njih sadrži podatke o jednoj klasi entiteta ili klasi poveznika iz realnog sistema. U ovom radu će fokus biti na tri SUBP-a – *Microsoft SQL Server*, *Oracle* i *PostgreSQL*.

4.1. Pregled SUBP-ova korišćenih u ovom radu

Sva tri korišćena SUBP-a u ovom radu – *Microsoft SQL Server*, *Oracle* i *PostgreSQL* predstavljaju sisteme za upravljanje relacionim bazama podataka. *Microsoft SQL Server* je podržan na *Linux* i *Windows* operativnim sistemima i dostupan je korisnicima u dve besplatne verzije – *Compact* i *Express*. Podržava proceduralna proširenja kroz postojanje *T-SQL* jezika. Osnovna razvojna okruženja koja se mogu koristiti za potrebe implementacije i održavanja baze podataka bazirane na *Microsoft SQL Server* SUBP-u su: *SQL Server Management studio*, *Azure Data Studio* i *Sql Server Data Tools*. *Oracle* SUBP je takođe podržan na *Windows* i *Linux* platformama.

U slučaju ovog SUBP, korisnici mogu birati između četiri različite verzije, a to su: *Enterprise edition*, *Standard edition*, *Express edition* (XE) i *Oracle lite*. Proceduralna proširenja su dostupna kroz postojanje *PL/SQL* jezika. Za implementaciju i korišćenje baze podataka upotrebom ovog SUBP-a najčešće korišćenja razvojna okruženja su *Sqldeveloper* i *SQL*plus*. Treći SUBP koji je korišćen u ovom radu, *PostgreSQL*, se dominantno koristi za implementaciju baza podataka na *MacOS* operativnim sistemima. Podržan je i na *Linux* i *Windows* platformama. Zasnovan je na klijent-server modelu i radi se o *open-source* SUBP-u. *PostgreSQL* takođe podržava proceduralna proširenja upotrebom jezika *PL/pgSQL* koji ima dosta sintaksnih i semantičkih sličnosti sa *PL/SQL* jezikom *Oracle* SUBP-a. Razvojna okruženja koja se mogu koristiti za implementaciju baze podataka upotrebom ovog SUBP-a su: *pgAdmin*, *phpPgAdmin* i *squirell*. S obzirom na to da su i *Oracle* i *PostgreSQL* i *Microsoft SQL Server* sistemi za upravljanje relacionim bazama podataka, nije čudno da postoji veliki broj karakteristika koje se mogu generalizovati na sva tri SUBP-a. Ipak, svaki od njih poseduje neke svojstvene odlike.

4.2. Tehnike indeksiranja u različitim SUBP

Indeksi baze podataka predstavljaju strukturu podataka čija namena je obezbeđenje brže pretrage i pronalaska podataka u okviru tabele baze podataka za koju su kreirani. Takođe, mogu se posmatrati i kao zona

podataka, sa kopijom podataka iz originalne tabele, u kojoj su podaci sortirani po određenom, logičkom, redosledu. Prilikom kreiranja indeksa bitno je voditi računa o tome da tabele za koje se kreira indeks zauzimaju više memorije i prostora u okviru baze podataka nego one koje nemaju definisane indekse, pa kreiranje nepotrebnih indeksa može iz ovog razloga dovesti do manje efikasnosti korišćenja prostora na eksternoj memoriji i povećanja vremena koje je SUBP-u potrebno za manipulaciju podacima. Prilikom kreiranja indeksa definiše se naziv indeksa, tabela nad kojom će biti kreiran i kolona (odnosno kolone) koja će se indeksirati. Ako nije drugačije naglašeno, podrazumeva se da se vrednosti indeksiraju u rastućem redosledu.

Svi analizirani sistemi za upravljanje bazama podataka podržavaju indekse bazirane na strukturi tipa B-stabla. S obzirom na to da oni podržavaju takva stabla u kojima se sve vrednosti indeksiranog obeležja pojavljuju u listovima stabla i da su listovi po pravilu spregnuti saglasno rastućoj ili opadajućoj vrednosti indeksiranog obeležja u daljem tekstu se navodi da se radi o B^+-stablu traženja, da bi se ukazalo na razliku u odnosu na osnovnu B-stablo traženja, kako je ono opisano u [5], kao i na činjenicu da se radi o nekoj modifikaciji osnovnog B-stabla.

Microsoft SQL Server podržava različite vrste indeksa među kojima su: grupisani indeksi (*clustered index*), negrupisani indeksi (*nonclustered index*) bez ili sa uključenim kolonoma, kolonski orijentisani indeksi (*columnstore index*), XML indeksi, filtrirani indeksi (*filtered index*), prostorni indeksi (*spatial index*) i tekst indeksi (*full-text index*) [7]. Dve najznačajnije vrste indeksa u *Microsoft SQL Server*-u, zasnovane na strukturi B^+-stabla – grupisani i negrupisani – su korišćene za testiranje u ovom radu. Grupisani indeks predstavlja strukturu tipa B^+-stabla u čijim listovima su smešteni svi podaci date torke. Najčešće se ova vrsta indeksa kreira nad obeležjem (ili nizom obeležja) koje predstavlja primarni ključ tabele i u jednoj tabeli se može kreirati najviše jedan indeks ove vrste.

Pri ažuriranju tabele nad kojom je kreirana ova vrsta indeksa automatski se vrši i ažuriranje redosleda u indeksnoj tabeli kako bi taj redosled bio usklađen sa novim fizičkim redosledom podataka u tabeli. Zbog činjenice da se u listovima grupisanog indeksa nalaze čitave torke a ne pokazivači ka lokaciji na kojoj su zapisane on se može istovremeno posmatrati i kao indeks i kao zona podataka. *Oracle* SUBP u sličnom kontekstu koristi *Index-Organized Tables* (IOT). Negrupisani indeks takođe predstavlja strukturu tipa B^+-stabla. Razlika u odnosu na grupisani indeks ogleda se u tome što u listovima nisu smeštene čitave torke već samo vrednosti obeležja po kojima je kreiran indeks i za svaku od njih pokazivač ka slogu u zoni podataka u kojem je zapisana torka sa tom vrednošću indeksnog obeležja. Upotreba ove vrste indeksa se pokazala kao korisna u situaciji kada je neophodno upitom izdvojiti relativno mali broj torki iz tabele koja sadrži veliki broj podataka [2].

Može se primetiti da iako se nalaze u fizički susednim elementima u listovima, pokazivači negrupisanog indeksa pokazuju na fizički razdvojene lokacije listova grupisanih indeksa. Ovo dalje ukazuje na to da pri postojanju negrupisanog indeksa nad kolonom, bez obzira na to što podaci u okviru nje nisu sortirani, nije neophodno pretražiti sve torke u tabeli u kojoj se ta kolona nalazi kako bi se pronašli željeni podaci, već se pomoću pokazivača ka listovima grupisanog indeksa direktno pristupa torkama čija vrednost primarnog ključa je tražena. *Oracle* SUBP takođe podržava jedinstvene i nejedinstvene indekse, kompozitne indekse (indekse nad složenim obeležjima), kompresovanje indeksnih vrednosti i druge različite tehnike indeksiranja. Neke od često korišćenih vrsta su: indeksi sa B^+-stablom, indeksno organizovane tabele (*Index Organized Table*, IOT), *bitmap* indeksi, *bitmap* spojeni indeksi i indeksi zasnovani na funkciji (*function-based* indeksi). Indeksi sa B^+-stablom su podrazumevano uređeni saglasno rastućoj vrednosti indeksiranih obeležja i kreiraju se pokretanjem naredbe *CREATE INDEX*. Indeksno organizovana tabela predstavlja B^+-stablo u kojem se u listovima stabla pored vrednosti indeksiranog obeležja (primarnog ključa) nalaze i vrednosti svih ostalih obeležja date torke.

Bitmap indeks se može posmatrati kao dvodimenzionalna mapa za vrednosti tabele, gde se kao redovi javljaju *rowID* tj. identifikatori reda tabele baze podataka, dok svaka kolona *bitmap* tabele predstavlja jednu od postojećih vrednosti u okviru kolone baze podataka nad kojom se kreira ova vrsta indeksa. Posmatra se presek kolone i reda bitmap tabele kako bi se utvrdilo da li za torku sa posmatranim *rowID*-jem postoji vrednost u posmatranoj koloni i ako je vrednost preseka 1 (*bit*) znači da postoji, dok se u suprotnom unosi 0. Indeksi zasnovani na funkciji se koriste za optimizaciju upita kod kojih se u *SELECT* naredbi javlja neki aritmetički izraz ili agregatna funkcija.

PostgreSQL obezbeđuje četiri vrste indeksa koji se mogu koristiti: B^+-stablo traženja, *hash* indeksi, GIN i GiST indeksi. Upotrebom *hash* indeksa se kao vrednosti indeksa ne čuvaju konkretne vrednosti podataka iz baze podataka, već se za svaku torku u tabeli definišu posebne vrednosti koje se javljaju kao rezultat *hash* funkcije, po kojoj su ovi indeksi i dobili naziv, a ovako izračunate vrednosti zauzimaju mali memorijski prostor. Može se desiti da, u slučaju postojanja velike količine podataka, dva različita podatka iz baze imaju istu *hash* vrednost [3]. GIN indeksi se koriste za indeksiranje kolona u kojima su smešteni podaci složenog tipa podataka. Njihova najznačajnija primena je u pretraživanju teksta. GiST indeksi generalizuju pojam B-stabla i na druge tipove podataka poput teksta, slike ili prostornih podataka, pa je jedna od najčešćih primena u geoprostornim bazama podataka.

5. PRIMENA TEORIJSKIH OSNOVA NA REALNOM PRIMERU

U nastavku je prikazan proces optimizacije konkretnog upita nad bazom podataka isprojektovanom za potrebe podrške poslovanju realnog sistema osiguravajuće kuće i prokomentarisani dobijeni rezultati. Baza podataka je za potrebe izrade praktičnog dela ovog rada implementirana pomoću tri različita relaciona SUBP – *Microsoft SQL Server*, *Oracle* i *PostgreSQL* i za svaki od njih je odabrano razvojno okruženje u okviru kog će se vršiti implementacija. To su respektivno: *Microsoft SQL Server Management Studio*, *Sqldeveloper* i *pgAdmin*. Svako od navedenih razvojnih okruženja ima integrisane alate za testiranje performansi upita putem generisanja planova izvršavanja upita i predlaganja izmena upita koje bi potencijalno mogle uticati na poboljšanje njegovih performansi.

U okviru tabele 1 prikazani su nazivi tabela baze podataka korišćene za testiranje i broj unetih torki u svaku od njih.

Tabela 1 – Broj unetih torki u okviru tabela baze podataka

Naziv tabele	Broj torki
BRANCH_OFFICE	20
EMPLOYEE	1000
INSURANCE_CLIENT	100.000
CONTRACTOR	80.000
INSURANCE_USER	50.000
CONTRACT	100.000
INSURANCE_POLICY	80.000
PAYMENT	240.000
INSURANCE_TYPE	4
RISK	6
INSURANCE_RISK	15

Za potrebe testiranja performansi odabrana su dva upita nad bazom podataka osiguravajuće kuće koji su prikazani na listinzima 1 i 2. Vreme koje SUBP utroši na selektovanje podataka traženih u ovim upitima je upotrebom alata integrisanih u okviru razvojnih okruženja ovih SUBP-ova izmereno pri inicijalnom izvršenju upita (bez upotrebe indeksa) kao i nakon kreiranja indeksa i ponovnog pokretanja upita.

Sva testiranja izvršena su na računaru sa *Windows* 10 operativnim sistemom. Procesor računara je 64-bit *Intel Core* i7 (6500U CPU, 2.50 GHz) sa 2 jezgra i 4 MB keš memorije, kapacitet instalirane operativne memorije 8 GB, disk *TOSHIBA* MQ01ABD100, kapaciteta 932 GB. Za potrebe testiranja su za sva tri SUBP-a implementirani indeksi tipa B^+-stabla traženja.

```
select first_name, last_name, occupation
from insurance_client
where first_name like 'P%';
```

Listing 1 – prvi slučaj upotrebe za testiranje performansi

```
select count(c.contract_num), first_name,
last_name, occupation
from contractor ct left join
insurance_client inc on
(ct.contractor_ID=inc.client_ID) left join
contract c on
(ct.contractor_ID=c.contractor_ID)
where first_name like 'P%'
group by first_name, last_name, occupation;
```

Listing 2 – drugi slučaj uoptrebe za testiranje performansi

U tabeli 2 su prikazana vremena utrošena na izvršenje upita za sva tri SUBP-a koji su predmet interesovanja u ovom radu, izražena u sekundama.

Tabela 2 – *Prikaz dobijenih rezultata testiranja*

Slučaj upotrebe		Vreme *Microsoft SQL Server*	Vreme *Oracle*	Vreme *PostgreSQL*
Prvi	bez indeksa	0.37 s	0.10 s	0.02 s
	sa indeksom	0.09 s	0.01 s	0.004 s
Drugi	bez indeksa	0.73 s	0.13 s	0.073 s
	sa indeksom	0.15 s	0.04 s	0.64 s

Iako je za oba slučaja upotrebe utvrđeno smanjenje vremena za tek nekoliko desetina sekundi kod svakog od tri posmatrana SUBP-a, to ne znači da je upotreba indeksa neefikasna jer je uzorak nad kojim je vršeno testiranje relativno mali pa se ne može sa sigurnošću reći da bi ostvarene prednosti u utrošenom vremenu od strane posmatranih SUBP-ova bile iste i kada bi se testiranje vršilo nad bazom podataka sa većom količinom podataka. Posmatranjem odnosa ostvarenog smanjenja u utrošenom vremenu i broja torki u okviru tabele *INSURANCE_CLIENT* moglo bi se zaključiti da je on proporcionalan jer bi svako povećanje broja torki u tabeli uticalo i na povećanje razlike u utrošenom vremenu za izvršavanje upita pre i nakon kreiranja indeksa, što dalje dovodi do smanjenja ukupnog vremena koje se utroši na izvršavanje upita.

6. ZAKLJUČAK

Rezultati praktičnog dela rada idu u prilog tvrdnji da korišćenje tehnika indeksiranja u velikoj meri može unaprediti proces obrade i optimizacije upita. Autor rada imao je prethodnog iskustva u radu samo sa razvojnim okruženjem *Sqldeveloper*, ali su i preostala dva razvojna okruženja bila relativno jednostavna za upotrebu i dobro dokumentovana, pa rad sa njima nije predstavljao problem. S obzirom na to da je testiranje izvršeno na skromnoj računarskoj konfiguraciji, koja nije bila namenski konfigurisana za potrebe testiranja, dobijeni rezultati se ipak ne mogu koristiti za donošenje opštijih zaključaka. U okviru daljih istraživanja testiranje bi trebalo sprovesti na snažnijoj računarskoj konfiguraciji, nad bazom podataka čije tabele sadrže mnogo veći broj torki, primenom većeg broja različitih upita i drugih indeksnih struktura koje u ovom radu nisu testirane.

7. LITERATURA

[1] Bal Gupta, S., & Mittal, A. (2017). *Introduction to Database Management System Second Edition, Kindle Edition.* Laxmi Publications Pvt Ltd.

[2] Cioloca, C., & Georgescu, M. (2011). Increasing Database Performance using Indexes. *Database Systems Journal* , 13-22.

[3] Crnko, N. (2018, march 01). *Korištenje hash-indeksa kod pretraživanja mysql baze podataka*. Retrieved september 22, 2019 from sistemac.srce.hr: https://sistemac.srce.hr/koristenje-hash-indeksa-kod-pretrazivanja-mysql-baze-podataka-123

[4] Lazarević, B., Marjanović, Z., & Aničić, N. (2008). *Baze podataka.* Beograd: Fakultet organizacionih nauka.

[5] Mogin, P. (2008). *Strukture podataka i organizacija datoteka.* Beograd: Računarski fakultet & CET.

[6] Mogin, P., Luković, I., & Govedarica, M. (2004). *Principi projektovanja baza podataka.* Novi Sad: Fakultet tehničkih nauka.

[7] Mukherjee, S. (2019). *Indexes in Microsoft SQL Server.* University of the Cumberlands.

Ivana Spasojević rođena je u Novom Sadu 1994. godine. Master rad iz oblasti Inženjerstvo informacionih sistema odbranila je 2019. godine.

kontakt: ivana.spasojevic@uns.ac.rs