Profiliranje i optimizacija baze podataka u sustavu SQL Server

Pekić, Lorena

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Sveučilišni prijediplomski studij računarstva

Završni rad

Profiliranje i optimizacija baze podataka u sustavu SQL Server

Rijeka, srpanj 2023.

Lorena Pekić
0069085538

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Sveučilišni prijediplomski studij računarstva

Završni rad

Profiliranje i optimizacija baze podataka u sustavu SQL Server

Mentor: izv.prof.dr.sc. Sandi Ljubić

Rijeka, srpanj 2023.

Lorena Pekić
0069085538

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
POVJERENSTVO ZA ZAVRŠNE ISPITE

Rijeka, 3. ožujka 2022.

Zavod: **Zavod za računarstvo**
Predmet: **Baze podataka**
Grana: **2.09.02 informacijski sustavi**

ZADATAK ZA ZAVRŠNI RAD

Pristupnik: **Lorena Pekić (0069085538)**
Studij: Preddiplomski sveučilišni studij računarstva

Zadatak: **Profiliranje i optimizacija baze podataka u sustavu SQL Server / Database Profiling and Optimization in SQL Server**

Opis zadatka:

Potrebno je razviti ogledni sustav s potpornom bazom podataka, uz demonstraciju mogućnosti i učinaka automatskog profiliranja i automatske optimizacije baze podataka u SUBP-u SQL Server. Klijentski dio oglednog sustava služi primarno za detekciju i vizualizaciju problema u radu s bazom podataka (npr. ispad sustava, sporo osvježavanje podataka, veliko vrijeme odziva, i slično). Na poslužiteljskoj strani sustava treba uvesti složenost u model baze podataka u kontekstu većeg broja tablica, veza raznih kardinalnosti, te dodavanjem primjerenog broja okidača i pohranjenih procedura. U takvome okruženju demonstrirati učinak modula za profiliranje baze podataka te za automatizirano pronalaženje mogućih poboljšanja. Argumentirati važnost indeksiranja, statistike, planova izvršavanja te estimirane cijene upita koji se koriste u procesu odabira najučinkovitijih operacija.

Rad mora biti napisan prema Uputama za pisanje diplomskih / završnih radova koje su objavljene na mrežnim stranicama studija.

Lorena Pekić
Zadatak uručen pristupniku: 21. ožujka 2022.

Mentor:

Doc. dr. sc. Sandi Ljubić

Predsjednik povjerenstva za završni ispit:

Prof. dr. sc. Kristijan Lenac

Izjava o samostalnoj izradi rada

Izjavljujem da sam samostalno izradila ovaj rad.

Rijeka, srpanj 2023.

_____

Lorena Pekić

Sadržaj

Popis slika

Popis tablica

Popis ispisa

Poglavlje 1

Uvod

U sklopu završnog rada, cilj je razviti ogledni sustav s potpornom bazom podataka, koji će demonstrirati mogućnosti i učinke automatskog profiliranja i optimizacije baze podataka u sustavu za upravljanje bazom podataka - Microsoft SQL Server (MSS). Fokus je na detekciji, vizualizaciji i analizi problema koji se mogu pojaviti prilikom rada s bazom podataka, poput ispadanja sustava, sporog osvježavanja podataka i dugog vremena odziva. Na poslužiteljskoj strani sustava, implementirana je složenost u model baze podataka kroz veći broj tablica, značajnom količinom podataka, različitim vezama između tablica te odgovarajućim brojem okidača. Kroz ovakvo okruženje prikazat će se učinak modula za profiliranje baze podataka te automatsko pronalaženje mogućih poboljšanja. Naglasak je na važnosti indeksiranja, statistike, planova izvršavanja te procijenjene cijene upita, koji su ključni faktori u odabiru najučinkovitijih operacija.

Implementirani sustav, koji simulira rad web trgovine, omogućit će dobivanje dubljeg uvida u proces optimizacije baze podataka i njegovu širu važnost unutar područja računarstva. Optimizacija baze podataka igra ključnu ulogu u osiguravanju učinkovitosti, brzine, skalabilnosti i pouzdanosti aplikacija.

Poglavlje 2

Opis sustava

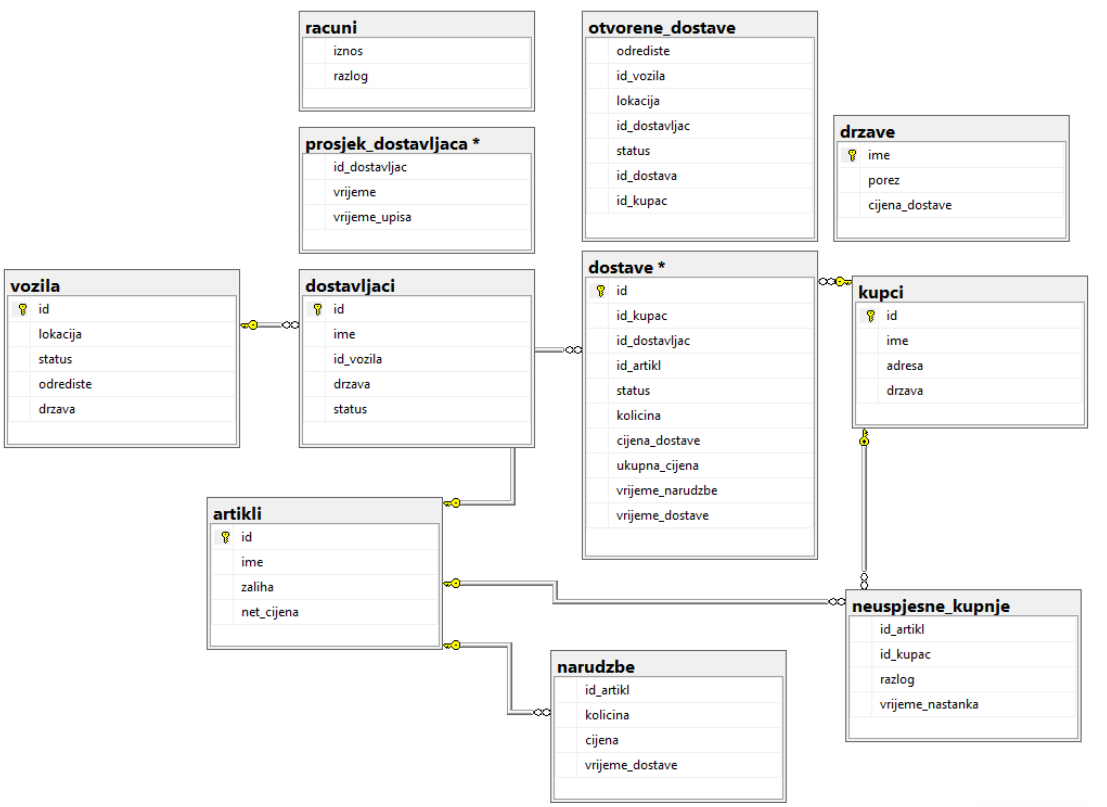
Razvijen je sustav zasnovan na potpornoj bazi podataka koji simulira rad međunarodne web trgovine koja obavlja dostave na zadanu adresu kupca. Sustav se sastoji od jedanaest tablica te ukupno trinaest okidača. Sustav je konstruiran na način da niz okidača provjerava različite stupce u različitim tablicama prije svakog unosa, ažuriranja ili brisanja podataka, kako bi se simuliralo veće opterećenje na sustav za upravljanje bazama podataka.

2.1 ER model

Na slici 2.1 prikazano je jedanaest tablica izgrađenog sustava pomoću MSS alata za stvaranje dijagrama baze podataka. Mogu se vidjeti povezanosti između tablica sa stranim ključevima te koje tablice imaju primarne ključeve (id). Uvedena struktura dućana funkcionira na način da se prilikom svakog unosa dostave, ukoliko ona uspješno prođe kroz sve okidače, dodaje u novu tablicu `otvorene_dostave`. U toj tablici se pomno prati status dostave, točnije udaljenost vozila dostavljača od odredišta, odnosno adrese kupca na dostavi. Kada udaljenost vozila dosegne vrijednost jednaku vrijednosti odredišta, računa se da je dostava izvršena te se njen status u tablici `dostave` mijenja u 'zatvorena', a dostavljač se postavlja kao 'slobodan' i dostupan za druge dostave. U tom trenutku se ta dostava briše iz tablice `otvorene_dostave` te se vrijeme potrebno dostavljaču da dostavi paket zapisuje

u tablicu `prosjek_dostavljaca`.

Kupci mogu na svojoj dostavi imati samo dostavljače iz svoje države, te se time cijena dostave izračunava na temelju cijene artikla i njegove količine, cijene dostave, te poreza za tu državu. Ukoliko dostava ne prođe jedan od okidača, ona se briše nakon unosa i razlog neuspjeha se zapisuje u tablicu `neuspjesne_kupnje`. Ako je zaliha nekog artikla ispod broja pet, nove zalihe se naručuju time da se unosi veći nasumičan broj za količinu naručene zalihe u tablicu `narudzbe`. Unosom u tu tablicu se unosi i dodatan redak u tablicu `racuni` koja predstavlja sve financijske gubitke dućana. U nju se ujedno može dodati i neki nasumičan unos (npr. štete, kazne, donacija...) sa nasumičnom vrijednosti iznosa.



Slika 2.1 *Grafički prikaz svih tablica u oglednoj bazi podataka*

2.2 Okidači

U ovom se poglavlju navode svi okidači i njihove funkcije u oglednom sustavu:

1. Provjera zalihe artikla

   Ako je unesena dostava čija količina artikla premašuje dostupnu zalihu artikla, onda se briše dostava i upisuje se u neuspješne kupnje.

2. Promjena zalihe nakon dostave

   Pri dodavanju nove dostave, broj zalihe artikla se smanjuje za broj količine artikla u dostavi. Ovaj okidač postavljen je kao zadnji okidač nakon unosa u tablicu dostava, kako bi se izbjegla nedosljednost podataka u tablicama.

3. Izračun ukupne cijene pri dostavi

   Prije samog dodavanja dostave, računa se ukupna cijena koja se postavlja na samu dostavu. Formula izračuna postavljena je na ispisu 2.1.

```
 SET ukupna_cijena = (@CijenaArtikla * @Kolicina )
+ @Porez/100 * (@CijenaArtikla * @Kolicina)
+ @CijenaDostave WHERE id = @IdDostave;}
```

Ispis 2.1 *Prikaz izračuna ukupne cijene na dostavi*

   Za ovaj izračun potrebno je dobaviti cijenu artikla na temelju njegovog primarnog ključa, potom državu u kojoj se obavlja dostava na temelju primarnog ključa kupca te naći informaciju o porezu i cijeni dostave.

4. Obnova zalihe artikla

   Nakon svakog ažuriranja tablice `dostave` provjerava se ako artikl na dostavi ima manje od pet dostupnih artikla, odnosno zalihe. U tom slučaju naručuje se artikl, točnije dodaje se nova narudžba u tablicu `narudzbe`.

5. Potrošnja na narudžbi

   Nakon upisa u tablicu `narudzbe`, što znači da se naručila nova zaliha artikla, upisuje se račun u tablicu `racuni` sa cijenom narudžbe i razlogom 'Narudžba

artikla'.

6. Provjera dostupnosti dostavljača u državi

   Ukoliko unesena dostava nema dostupnih dostavljača u toj državi, otkazuje se unos dostave i upisuje se u tablicu `neuspjesne_kupnje`.

7. Zauzet status dostavljača

   Nakon unosa dostave, ako je status dostave otvoren, postavlja se status dostavljača na dostavi u 'zauzet'.

8. Novi prosjek dostavljača

   Ako je dostava u statusu 'zatvorena' i ima stupac `vrijemeDostave` na sebi postavljen, onda se ubacuje primarni ključ dostavljača, razlika između vremena dostave (kada je dostava unesena u sustav) i vrijeme narudžbe (kada je dostava isporučena ili zatvorena), te samo vrijeme dostave kao vrijeme upisa u tablicu `prosjek_dostavljaca`.

9. Unos uspješnih otvorenih dostava u novu tablicu

   Ako se uspješno unese dostava koja je otvorenog statusa, unose se potrebni podaci u vezi dostave u tablicu `otvorene_dostave`. To uključuje direktne podatke o lokaciji vozila i odredištu koje tablica `dostave` nema.

10. Provjera lokacije vozila

    Ako se u tablici `otvorene_dostave` ažurira dostava tako da je sada njeno odredište jednako vrijednosti trenutne lokacije, onda se zatvara dotična dostava tako da joj se postavlja status u 'zatvorena'. Ovaj okidač će potom okinuti sljedeće zatvaranje dostave.

11. Zatvaranje dostave

    Ako se ažurira dostava u tablici `otvorene_dostave` tako da joj je novi status 'zatvorena', onda se također ažurira taj status u tablici `dostave` za dostavu s istim primarnim ključem i briše se dotična dostava iz otvorenih dostava. Ovaj okidač će ujedno inicijalizirati početnu lokaciju vozila (vrijednost nula).

12. Slobodan status dostavljača

    Ako je dostava prešla u status 'zatvorena', onda se postavlja status njenog

dostavljača u status 'slobodan'.

13. Brisanje dostave

    Kada se briše neki redak u dostavi, ovaj okidač umjesto brisanja prvo vraća status dostavljača natrag u 'slobodan', te vraća količinu artikla sa dostave natrag u zalihu artikla. Potom se izvrši samo brisanje dostave.

2.3 Unos podataka

Uspostavljena je jedna pregledna web stranica pod nazivom `naslovnica.php` u kojoj se nalaze poveznice na ostale stranice za upis i ispis podataka. Za potrebu ovog rada su se upisi uglavnom vršili putem izvršavanja php skripti kroz terminal, pošto se radi o periodičnom unosu nasumičnih podataka. Za primjer na ispisu 2.2 prikazan je upis dostava gdje je više informacija potrebno: primarni ključ postojećeg kupca, artikla i slobodnog dostavljača. U programu ostvaruju se upiti prema tri tablice (`artikli`, `kupci` i `dostavljaci`) te se potom unosi novi redak u tablicu `dostave`. Unos se obavlja svakih tri sekunde, te se radi preglednosti odgovarajući Structured Query Language (SQL) upit ispisuje na stranicu ili terminal.

Pri samom početku skripte `unosDostave.php` uključuje se `dbh.php` stranica gdje se uspostavlja veza sa poslužiteljem i specifičnom bazom (u ovom slučaju baza podataka s imenom `ducan`). Na ispisu 2.3 prikazana je funkcija kojom se dobiva prvi nasumični kupac, te se vraća njegov primarni ključ `id` i stupac `drzava`.

Količina artikla se dobiva koristeći Box-Mullerovu transformaciju za generiranje nasumičnih brojeva na temelju normalne distribucije s navedenom srednjom i standardnom devijacijom. *Do-while* petlja osigurava da generirani broj bude unutar željenog raspona [1]. Na taj način jamči se veća šansa da će se izbjeći unos dostave koja traži veću količinu artikla od njegove dostupne zalihe.

Kako bi se omogućio što veći broj okidača, većina implementacijske logike se smjestila u okidače na samoj bazi podataka umjesto u program poslužiteljske strane prije samog unosa u tablicu. Stoga se u prikazanoj funkciji uglavnom upisuju prvi dobiveni nasumični kupac, artikl i dostavljač, bez obzira na to je li količina na dostavi veća od zalihe artikla i slično. Tek pri unosu će se proći kroz sve važne kriterije koji

```
$buyer = get_buyer();
$freeDrivers = get_driver($buyer[2]);
$article = get_article();
if ($buyer != -1 && $freeDrivers != -1 && $article != -1) {
    $quantity = randomNumber(50, 10, 1, 5000);
    $params = array();
    $params[1] = $buyer[1];
    $params[2] = $freeDrivers;
    $params[3] = $article;
    $params[4] = 'otvorena';
    $params[5] = $quantity;
    $params[6] = date("Y-m-d");
     $result = "INSERT␣INTO␣dostave␣(id_kupac,id_dostavljac,
        id_artikl,status,kolicina,vrijeme_narudzbe)␣VALUES␣(" . $
        buyer[1] . "," . $freeDrivers . "," . $article . ",'otvorena
        '," . $quantity . "," . $currentDate . ")"
    $stmt = sqlsrv_query($conn, $result, $params);
    if ($stmt) {
    } else {
        echo "error";
        die(print_r(sqlsrv_errors(), true));
    }
}
```

Ispis 2.2 *Prikaz funkcije za unos novih podataka u tablicu* `dostave`

će odrediti ako je dotična dostava zapravo uspješna ili nije. Po unosu dostave se također aktiviraju okidači koji mijenjaju stupce u drugim tablicama. Primjerice, ako je prošla dostava, uzet će se dostavljač na njoj i promijeniti će se njegov status iz 'slobodan' u 'zauzet'. Kako bi se izbjegli problemi s kaskadnim promjenama, kada aktivacija nekog okidača mijenja drugu tablicu, ali sama dostava na kraju biva neuspješna, uvedeni su dodatni okidači koji pri brisanju neuspješne dostave dodaju dostavu u novu tablicu `neuspjesne_kupnje` te vraćaju sve potrebne informacije na povezane tablice (status dostavljača, status i lokacija vozila, te zaliha artikla).

```
function get_buyer(){
    include 'dbh.php';
    $sql = "SELECT␣TOP␣1␣*␣FROM␣kupci␣ORDER␣BY␣NEWID();";
    $params = array();
    $options = array("Scrollable" => SQLSRV_CURSOR_STATIC);
    $stmt = sqlsrv_query($conn, $sql, $params, $options);
    $row_count = sqlsrv_num_rows($stmt);
    if($row_count == 0){
        return -1;
    } else {
        $row = sqlsrv_fetch_array($stmt, SQLSRV_FETCH_ASSOC);
        if($row){
            $params = array();
            $params[1] = $row['id'];
            $params[2] = $row['drzava'];
            return $params;
        }
    }
}
```

Ispis 2.3 *Implementacija funkcije* `get_buyer()` *koja traži nasumičnog kupca za dostavu*

Poglavlje 3

Savjetnik za optimizaciju baze podataka

Savjetnik za optimizaciju MSS baze podataka (eng. Microsoft Engine Tuning Advisor (META)) je alat za optimizaciju izvedbe sa sposobnostima analize i preporuka. Ovaj alat pomaže programerima i administratorima identificirati i riješiti probleme sa izvođenjem sustava, što rezultira poboljšanim vremenom odziva i učinkovitosti infrastrukture. Prvi put je predstavljen kao dio Microsoft SQL Servera 2000. Tijekom godina Microsoft je kontinuirano poboljšavao algoritme savjetnika i proširivao njegove mogućnosti, čineći ga vrijednim alatom za optimizaciju izvedbe upita.

3.1 Princip korištenja

Primarna svrha savjetnika za optimizaciju je analizirati obrasce radnog opterećenja (eng. *workload*) na bazi podataka SQL Servera i dati preporuke za optimiziranje planova izvršavanja upita. Uzima u obzir čimbenike kao što su struktura tablice, indeksi i statistike kako bi se identificirala potencijalna područja poboljšanja. Osim identificiranja prilika za poboljšanje, savjetnik za optimizaciju baze podataka će također izraditi T-SQL skriptu koja se može pokrenuti kako bi se implementirale njegove preporuke. Savjetnik se obično koristi u sljedećim scenarijima:

- Podešavanje izvođenja

Analizirajući uzorke upita i predlažući promjene indeksa, pomaže optimizirati izvršenje upita i smanjiti vrijeme odziva.

- Optimizacija upita tijekom razvoja

  Razvojni programeri mogu koristiti savjetnika tijekom faze razvoja aplikacije za podešavanje upita i poboljšanje njihove izvedbe. Analizom različitih alternativa upita i strategija indeksiranja, pomaže u odabiru najučinkovitijeg plana izvršavanja.

- Planiranje kapaciteta

  Savjetnik pomaže u procjeni zahtjeva za resursima analizirajući radno opterećenje i predlažući fizičke konfiguracije. Pomaže osigurati da se infrastruktura baze podataka može učinkovito nositi s predviđenim radnim opterećenjem.

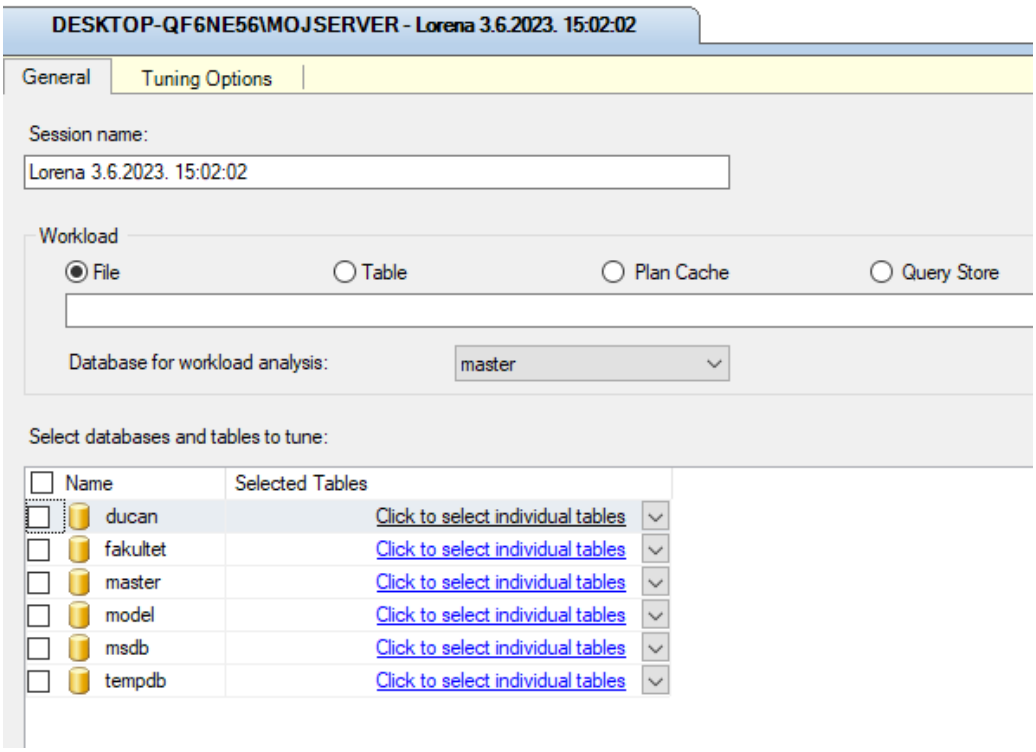
Nakon pokretanja savjetnika kroz izbornik Alati, otvara se novi prozor gdje se stvara sjednica za optimiziranje. Osim imena sjednica i liste dostupnih baza na povezanom poslužitelju, u prozoru se pojavljuje i izbor radnog opterećenja koji će se analizirati, što je prikazano na slici 3.1.

Radno opterećenje je skup *Transact-SQL* naredbi koje se izvršavaju nad jednom ili više baza podataka koje se žele prilagoditi. Savjetnik prihvaća reprezentativno radno opterećenje profiliranjem izvršenja uzorka upita na ciljanoj bazi podataka [2]. Ovi podaci o radnom opterećenju uključuju informacije o izvršenim upitima, njihovoj učestalosti i povezanoj potrošnji resursa.

Radno opterećenje može se manualno napraviti kroz uređivač upita u MSS studiju ili uređivačem teksta tako da se upisane SQL naredbe spreme kao .sql datoteka. Potom se u savjetniku odabere opcija Datoteka (eng. *File*) da bi se pretražila i postavila stvorena .sql datoteka.

Drugi način predaje radnog opterećenja savjetniku je kroz predmemoriju plana (eng. *Plan Cache*) gdje savjetnik uzima podskup 1000 nedavno izvršenih upita. Ti su podaci spremljeni u predmemoriji sustava, stoga se izgube kada se ponovno pokrene (eng. *reset*) poslužitelj.

S druge strane, opcija korištenja dućana upita (eng. *Query Store*) ima u sebi spremljene upite koji su se izvršavali nad bazom podataka kroz dulji period vremena,

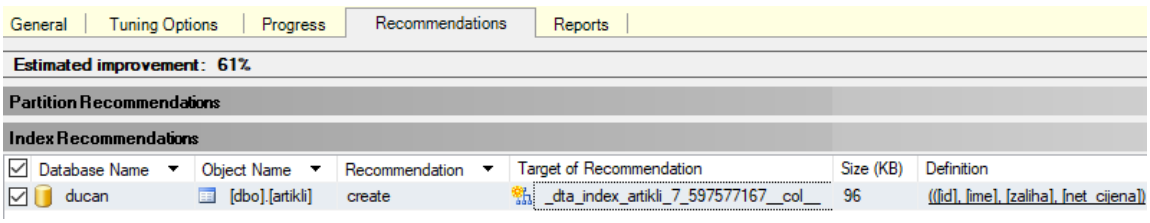


Slika 3.1 *Prikaz prozora gdje se stvaraju sjednice za savjetnika za optimizaciju MSS baze podataka*

bez obzira na ponovno pokretanje poslužitelja.

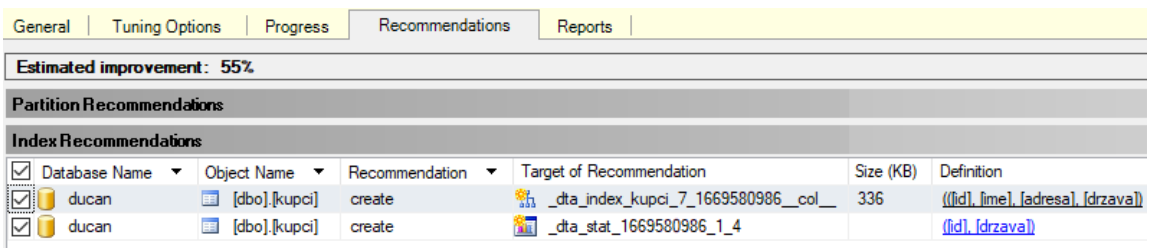
3.2 Primjer optimizacije

U demonstraciji će se koristiti .sql datoteku upita koji vraća deset kupaca sa najvećom ukupnom potrošnjom na svojim dostavama te dodatno računa prosječnu potrošnju po svakom kupljenom artiklu. Nakon analize, savjetnik za optimizaciju baze podataka predstavlja preporuke za indekse koji su prezentirani na slici 3.2. Procijenjeno poboljšanje na operativni trošak upita iznosi čak 61%, samo stvaranjem negrupiranog indeksa u tablici artikala.



Slika 3.2 *Prikaz savjeta za indekse od Microsoft savjetnika za optimizaciju baze podataka*

Na slici 3.3 je primjer drugog upita koji rezultira sa dvije preporuke: negrupirani indeks na tablici kupaca te statistike. Implementacijom navedenih preporuka bi se ubrzalo izvršenje upita za 55%. Na listi preporuka prikazani su podaci o indeksima koji bi se stvorili: njihova veličina, nad kojom tablicom i kojim stupcima u toj tablici bi se stvorili. Predstavljen je izbor preporuka koje korisnik želi primijeniti na svoj sustav. Moguće je zakazati automatsku implementaciju savjeta te spremiti T-SQL naredbe za stvaranje indeksa/statistika za kasnije provođenje.



Slika 3.3 *Prikaz savjeta za uvođenje indeksa i statistika*

3.3 Nedostaci

Savjetnik izvodi pojednostavljenu analizu na temelju dostavljenog radnog opterećenja, koje možda neće pokriti sve moguće scenarije upita. Alat može propustiti određene uzorke upita ili propustiti uzeti u obzir određene rubne slučajeve. Na primjer, za pobošljanje izvršavanja jednog upita META alat može preporučiti izradu jednog indeksa, ali kasnije za poboljšanje drugog upita može preporučiti brisanje tog istog indeksa. Optimizacija funkcionira na razini danog upita, tako da njegovi savjeti ne mogu biti globalno optimalni u svim slučajevima. U nekim slučajevima, savjetnik može preporučiti pretjerano indeksiranje, što dovodi do nepotrebnih troškova i povećanog održavanja. Presudno je pregledati i potvrditi preporuke kako bi bili sigurni da su usklađene sa specifičnim zahtjevima i ograničenjima sustava. Savjetnik sadrži restrikcije i ograničenja, kao što su nemogućnost dodavanja ili brisanja jedinstvenih indeksa ili indeksa koji provode `PRIMARY KEY` ili `UNIQUE` ograničenja. Ne može analizirati bazu podataka koja je postavljena na način rada za jednog korisnika. Ako su pri izradi sjednice dodana ograničenja za maksimalni prostor na disku za preporuke podešavanja (pomoću dijaloškog okvira Napredne opcije podešavanja), META alat može biti prisiljen izbrisati određene postojeće indekse. U takvome slučaju, rezultirajuća preporuka savjetnika može proizvesti negativni rezultat pri optimizaciji.

Poglavlje 4

Optimizator upita

Kako bi mogao izvršavati upite, MSS mora prvo analizirati danu izjavu kako bi odredio učinkovit način pristupa potrebnim podacima i njihovu obradu. Ovom analizom upravlja komponenta koja se zove optimizator upita (eng. *Query Optimizer*). Unos u optimizator upita sastoji se od upita, sheme baze podataka (definicije tablice i indeksa) i statistike baze podataka. Optimizator gradi jedan ili više planova izvršavanja upita, koji se ponekad nazivaju planovi upita ili planovi izvršavanja. Plan izvršavanja je skup uputa koje opisuju koji se koraci procesa izvode dok jezgra sustava za upravljanje bazom podataka izvršava upit. Optimizator procjenjuje trošak potrošnje resursa kandidata za plan upita i pokušava odabrati učinkovit i jeftin plan u smislu I/O (ulaz/izlaz - eng. *input/output*), memorije, centralne procesne jedinice (eng. *Central Processing Unit (CPU)*) i vremena kako bi uravnotežio vrijeme kompilacije i optimalnost plana [3].

Optimizator upita koristi sljedeće tehnike pri optimizaciji upita kako bi skovao procijenjeni plan izvršavanja:

- Otkrivanje proturječnosti i ograničenja provjere
- Pojednostavljenje
- Eliminacija pridruživanja

  Eliminacija pridruživanja (eng. `JOIN`) je tehnika kojom se pojednostavljuju upiti tako da se identificiraju nepotrebna pridruživanja tablica te se ista elimi-

niraju sa ciljem poboljšanja učinkovitosti izvršavanja upita.

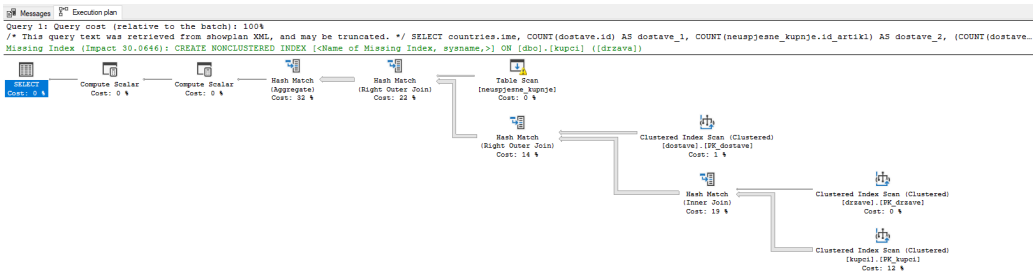
- Cilj redaka

  Optimizator upita procjenjuje koliko redaka će upit morati vratiti i na temelju te informacije izračunava potrošnju resursa za upite. Optimizator primjenjuje ovu strategiju kada se neka od ključnih riječi koje definiraju broj redaka (npr. `TOP`, `SET ROWCOUNT`, `FAST`, `IN`, `EXISTS`...) pojavljuje u upitu. Tijekom ove strategije procijenjeni broj redaka će biti smanjen što rezultira manjom potrošnjom resursa. Procijenjeni plan može se razlikovati od stvarnog plana izvršavanja, a glavni razlog je stara, odnosno neažurirana statistika. Optimizator upita koristi statistiku za izradu ekonomičnog i optimiziranog plana izvršavanja upita, stoga ako statistika nije ažurirana ili ako uzorkovanje podataka nije odgovarajuće, alat za optimizaciju upita izvršit će potpunu optimizaciju i izraditi plan izvršavanja na temelju starih informacija. Zastarjela statistika ponekad se pojavljuje kada se podaci često mijenjaju i statistika se ne ažurira u skladu s održavanjem indeksa [4].

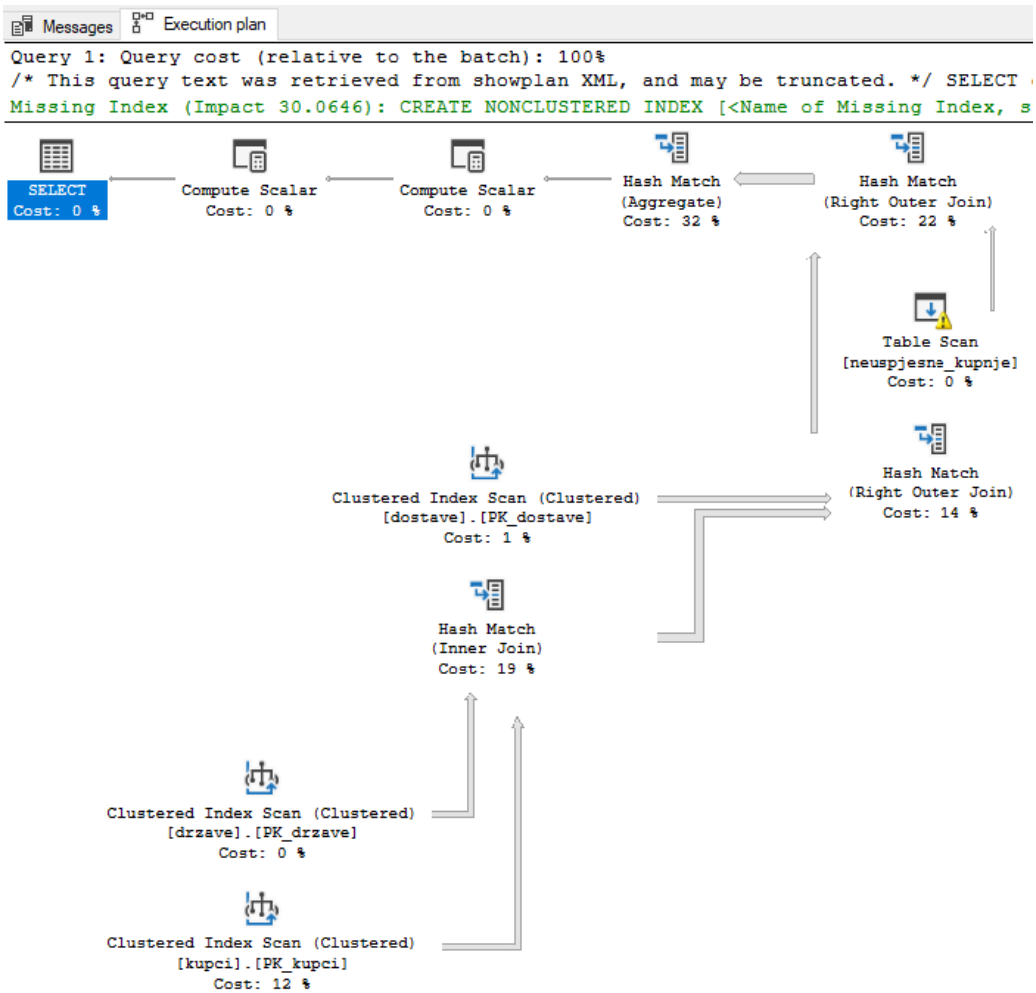
4.1 Demonstracija procijenjenog plana izvršavanja

Na slici 4.1 prikazan je procijenjeni plan izvršavanja za upit koji za svaku državu u tablici dohvaća i izračunava postotak uspješnih dostava nad ukupnim brojem dostava.

U planu izvršavanja redom se izvršavaju sljedeći upiti: `SELECT` koji dohvaća retke, `SORT` koji potom te retke sortira silaznim redom prema ukupnom trošku na dostavi, te izračunavanje (eng. *compute scalar*) koje podrazumijeva izvršavanje odgovarajućih matematičkih operacija nad dobivenim podacima. Potom se agregiraju retci u tablicu i na kraju se izvršavaju dvije `JOIN` operacije koje skeniraju kroz grupirane indekse (primarne ključeve artikla, kupca i dostave) te dva trenutno postojeća negrupirana indeksa u artiklima i kupcima. Tijekom optimizacije upita, optimizator prolazi kroz svaku operaciju i



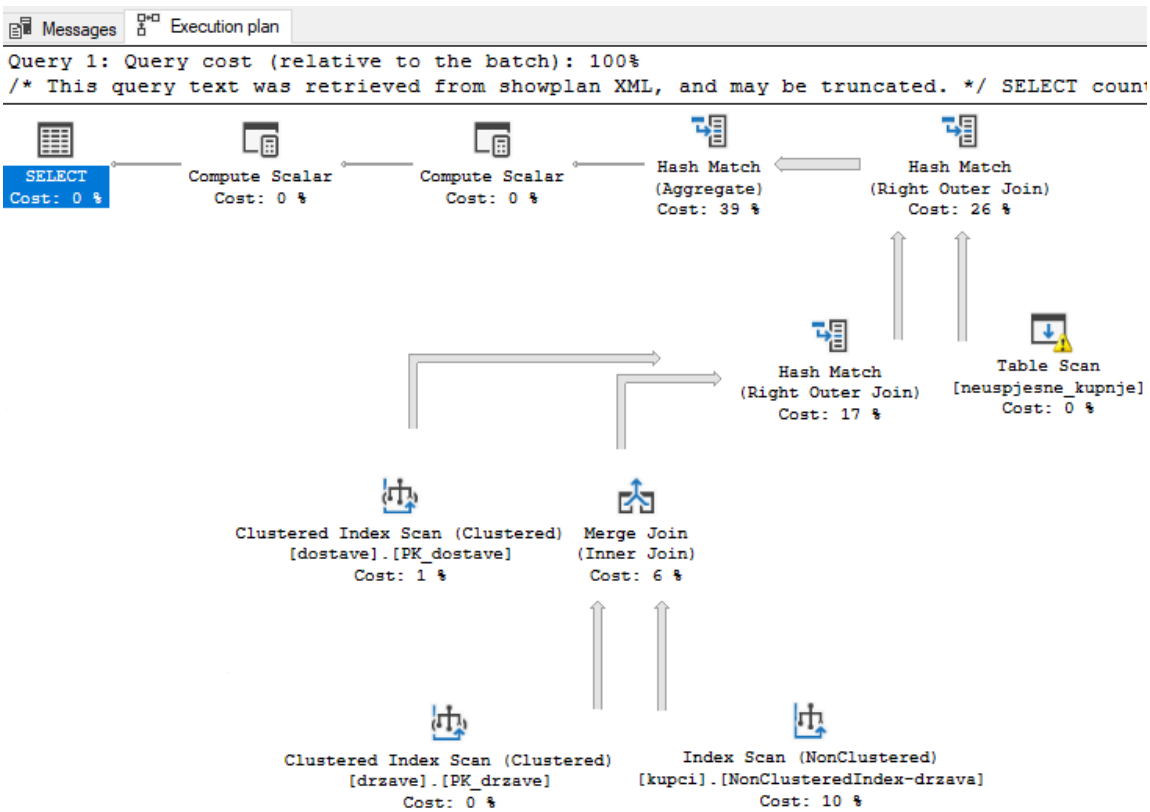
4.1.1 *Originalni prikaz procijenjenog plana izvršavanja*



4.1.2 *Uređeni prikaz procijenjenog plana izvršavanja radi detaljnijeg pregleda*

Slika 4.1 *Prikaz procijenjenog plana izvršavanja na temelju upita*

dodijeljuje joj trošak. Trošak operacije je relativna mjera koja se odnosi na procijenjeni računalni napor potreban za izvođenje određene operacije unutar plana upita. Njegova vrijednost se izračunava dijeljenjem procijenjenog troška operatera s procijenjenim troškom podstabla gornjeg lijevog operatora (što je ukupni kombinirani procijenjeni trošak jednog operatera nad svim operaterima u upitu). Cilj optimizatora je da odabere plan izvršavanja sa najmanjim troškom izvršavanja operacija. Zbrajanjem svih postotaka u planu dobije se 100%. Zelenim tekstom iznad grafičkog prikaza je tekst koji obavještava korisnika da mu "*nedostaje*" negrupirani indeks na tablici `kupci` za stupac `drzave`. Nakon implementiranja istaknutog indeksa dobije se novi procijenjeni plan izvršavanja na slici 4.2. Zbrajanjem novih postotaka u planu se dobije 99%, što znači da je unos negrupiranog indeksa poboljšao trošak operacije za 1%.



Slika 4.2 *Prikaz procijenjenog plana izvršavanja nakon primjene preporučenog indeksa*

Može se vidjeti da su skeniranja nad negrupiranim indeksima tipično manjeg operativnog troška nego nad grupiranim indeksima. Neki od razloga su ti što negrupirani indeksi obično sadrže samo indeksirane stupce i referencu na grupirani indeks ili stvarne retke podataka. Ova uža struktura omogućuje kompaktniju pohranu i učinkovito skeniranje indeksnih stranica. Negrupirani indeksi općenito su manje veličine u usporedbi s grupiranim indeksima. Manji indeks zahtijeva manje I/O operacija diska za čitanje i prelazak svih njegovih stranica tijekom skeniranja. Ovo smanjenje I/O operacija doprinosi nižim ukupnim troškovima skeniranja. Negrupirane indekse može se skenirati bilo kojim redoslijedom na temelju predikata pretraživanja, te često imaju i pliću hijerarhiju koja zahtijeva manje razina za navigaciju.

Poglavlje 5

Indeksiranje i statistike

Indeksi i statistike predstavljaju ključne elemente u optimizaciji izvođenja baza podataka. Oni pružaju efikasno pretraživanje, filtriranje i sortiranje podataka, čime značajno poboljšavaju vrijeme izvršavanja upita. Indeksi omogućuju brzi pristup podacima, dok statistike pružaju informacije o raspodjeli podataka u tablicama.

5.1 Indeksiranje

Indeks je struktura podataka povezana s tablicom baze podataka koja omogućuje učinkovito dohvaćanje, filtriranje i sortiranje podataka na temelju određenih stupaca ili polja. Djeluje kao pokazivač koji pomaže mehanizmu baze podataka da brzo locira željene podatke, bez potrebe za skeniranjem cijele tablice. Indeks se stvara na jednom ili više stupaca tablice i sadrži sortiranu kopiju indeksiranih stupaca zajedno s referencom na odgovarajuće retke u tablici.

Prednosti korištenja indeksa u bazi podataka su višestruke. Stvaranjem indeksa na stupcima koji se često traže rezultira bržim vremenom odziva i poboljšanom ukupnom propusnošću sustava. Indeksi također poboljšavaju integritet podataka nametanjem jedinstvenosti i ograničenja primarnog ključa. Oni osiguravaju da se duplikati ili nevažeći podaci ne mogu umetnuti u indeksirane stupce, čime se održava integritet i dosljednost baze podataka.

Indeksiranje zahtijeva temeljito razumijevanje podataka, obrazaca upita i zah-

tjeva za izvedbu kako bi se stvorili odgovarajući indeksi koji maksimiziraju izvedbu dok minimaliziraju troškove pohrane i održavanja. Redoviti nadzor i fino podešavanje indeksa potrebni su kako bi se osigurala optimalna izvedba kako se baza podataka razvija tijekom vremena.

5.1.1 Tipovi indeksa

- **Grupirani indeksi**

  Grupirani (eng. *Clustered*) indeksi određuju fizički poredak podataka u tablici time što pohranjuju podatke o redoslijedu. Grupirani indeksi imaju ključnu vrijednost (eng. *key value*), što znači da ne mogu imati duplicirane vrijednosti. Jedna tablica može imati samo jedan grupirani indeks, a svaki takav indeks će u sebi sadržavati pokazivač na red podataka u tablici, ali ne i na određeni stupac. Ako se tablici dodaje primarni ključ, ona automatski postaje grupirani indeks [5]. Posebno su korisni za upite temeljene na rasponu, odnosno podacima razvrstanima po rastućem ili opadajućem redoslijedu.

- **Negrupirani indeksi**

  Negrupirani (eng. *Non-clustered*) indeksi su zasebne strukture koje sadrže kopiju indeksiranih stupaca i referencu na odgovarajuće podatke. Kada se postavi upit za stupac na kojem je stvoren negrupirani indeks, baza podataka će prvo otići do tog indeksa i potražiti adresu odgovarajućeg retka u tablici. Zatim će otići na adresu tog retka i dohvatiti druge željene vrijednosti stupaca [6]. U negrupiranim indeksima podaci su raspoređeni nasumično, ali logički redoslijed interno je određen indeksom. Jedna tablica može imati više negrupiranih indeksa. Negrupirani indeksi osobito dobro funkcioniraju sa tablicama u kojima se podaci često mijenjaju [7].

5.1.2 Nedostaci

Usprkos mnogih prednosti indeksiranja tablica u bazi podataka postoje i nedostaci koje treba uzeti u obzir prije unošenja indeksa. Negrupirani indeksi troše dodatni prostor na disku jer pohranjuju dodatne kopije indeksiranih podataka. Prostor ovisi

o različitim čimbenicima koji se koriste u indeksu, poput veličine tablice, broja stupaca i vrste stupaca. Prilikom obrade baze podataka s velikim brojem korisnika, diskovni prostor se smatra jeftinijim od izvedbe aplikacije. Indeksi imaju loš utjecaj na izvedbu upita za modifikaciju podataka kao što su umetanje, ažuriranje ili brisanje. Svaki put kada upit traži izmjenu podataka u tablici, baza podataka se sama ažurira s novim indeksom u kojem se podaci mijenjaju. Kao što je ranije spomenuto, indeksi pomažu brže locirati zapise, što dovodi do bržeg sortiranja i pretraživanja. Stoga, previše indeksa može pomoći da se brže pronađu zapisi, ali slabo utječe na brzinu izmjene podataka [8]. Dakle, potrebno je imati dobar broj indeksa koji uravnotežuje učinak sustava. Potrebno je pažljivo razmotriti odabir odgovarajućih stupaca za indeksiranje kako bi se uspostavila ravnoteža između izvedbe upita i troškova održavanja indeksa.

5.2 Statistike

U MSS studiju, statistika pruža vrijedne informacije o distribuciji i karakteristikama podataka unutar tablice ili indeksiranog prikaza. Oni pomažu optimizatoru upita donositi informirane odluke o planovima izvršavanja upita i poboljšati izvedbu upita.

Kada se stvara indeks u tablici, SQL Server automatski stvara statistiku za taj indeks. Statistika se također može stvoriti ručno korištenjem naredbe `CREATE STATISTICS`. Statistika se sastoji od koraka histograma, informacija o gustoći i dodatnih statističkih metapodataka.

Komponente koje statistike sadrže:

- **Histogrami**

  Statistike SQL poslužitelja pohranjuju distribuciju podataka stupca u histogram, a omjer jedinstvenih vrijednosti u vektor gustoće. Ova dva meta-podatka koristi optimizator upita za izračunavanje koliko redaka će vratiti upit. Histogrami predstavljaju učestalost vrijednosti skupa podataka stupca. Alat za optimizaciju upita izračunava histogram na vrijednostima stupaca u prvom ključnom stupcu objekta statistike, odabirom vrijednosti stupaca statističkim uzorkovanjem redaka ili izvođenjem potpunog skeniranja svih redaka u tablici

ili prikazu. Za izradu histograma, optimizator upita razvrstava vrijednosti stupaca, izračunava broj vrijednosti koje odgovaraju svakoj zasebnoj vrijednosti stupca, a zatim agregira vrijednosti stupaca u najviše 200 uzastopnih koraka histograma.

- **Vektor gustoće**

  Vektor gustoće je informacija o broju duplikata u određenom stupcu ili kombinaciji stupaca i izračunava se kao 1/(broj različitih vrijednosti). Alat za optimizaciju upita koristi vektor gustoće za poboljšanje procjena kardinalnosti za upite koji vraćaju više stupaca iz iste tablice ili indeksiranog prikaza. Kako se gustoća smanjuje, selektivnost vrijednosti raste [9].

- **Filtrirane statistike**

  Filtrirane statistike mogu poboljšati izvedbu upita za upite koji biraju iz dobro definiranih podskupova podataka. Filtrirana statistika koristi predikat filtra za odabir podskupa podataka koji je uključen u statistiku. Dobro dizajnirana filtrirana statistika može poboljšati plan izvršavanja upita u usporedbi sa statistikom cijele tablice.

Optimizator upita koristi statistiku za procjenu kardinalnosti (broj redaka) i distribucije vrijednosti u tablici. Ove informacije pomažu optimizatoru odabrati najučinkovitiji plan izvršavanja za upit. Precizna statistika omogućuje optimizatoru generiranje optimalnih planova upita, što dovodi do poboljšane izvedbe upita. Statistika daje bitne podatke za model temeljen na troškovima optimizatora upita, pomažući mu u procjeni troškova različitih planova upita. Optimizator koristi statistiku za usporedbu različitih alternativnih planova i odabir plana s najnižim procijenjenim troškom.

Statistika igra ključnu ulogu u određivanju hoće li se indeks koristiti ili ne. Optimizator analizira statistiku kako bi procijenio selektivnost indeksa. Ako statistika pokazuje da će indeks značajno smanjiti broj redaka za obradu, veća je vjerojatnost da će optimizator iskoristiti taj indeks. Kada dođe do značajne promjene u temeljnim podacima, SQL Server može automatski ažurirati statistiku povezanu s pogođenim stupcima. Ovo pokreće rekompilaciju plana upita, omogućujući optimizatoru da se prilagodi novim statističkim informacijama i generira učinkovitije planove.

Na slici 5.1 prikazana su svojstva jedne stvorene statistike za stupac `ime` u tablici `artikli`. Mogu se vidjeti neke od prethodno spomenutih komponenata kao što su koraci histograma i vektor gustoće. U MSS-u može se pregledavati i analizirati statistika kako bi se dobio uvid u distribuciju podataka, korištenje indeksa i izvedbu upita. Ove informacije mogu pomoći u prepoznavanju uskih grla u izvođenju, zastarjelih statistika ili statistika koje nedostaju te pomoći u rješavanju problema i optimizaciji upita. Općenito, statistika daje bitne informacije alatu za optimizaciju upita, omogućujući mu donošenje inteligentnih odluka u stvaranju planova upita. Redovito ažuriranje i održavanje statistike presudno je za osiguranje optimalne izvedbe baze podataka SQL Servera.

Table Name: dbo.artikli

Statistics Name: _WA_Sys_00000002_239E4DCF

Statistics for INDEX '_WA_Sys_00000002_239E4DCF'.

Name	Updated	Rows	Rows Sampled	Steps	Density	Average Key Length
_WA_Sys_00000002_239E4DCF	Feb 18 2023 8:07PM	22381	22381	195	1	10

All Density	Average Length	Columns
4.468076E-05	10	ime

Histogram Steps

RANGE_HI_KEY	RANGE_ROWS	EQ_ROWS	DISTINCT_RANGE_ROWS	AVG_RANGE_ROWS
00BCEMqkPh	0	1	0	1
0eFoFcZv2l	94	1	94	1
0ncSjU7ow6	103	1	103	1
0s8Gvs23ZL	63	1	63	1
14V6c7gbpo	127	1	127	1
1cy8gDkz9u	63	1	63	1

Slika 5.1 *Prikaz svojstva statistike za stupac `ime` u tablici `artikli`*

Poglavlje 6

Eksperiment

Kako bi se dokazao učinak alata za optimizaciju baze podataka uspostavljen je eksperiment koji će mjeriti prosječno vrijeme odziva pojedinih upita u bazu podataka. Jedno mjerenje obavit će se nad inicijalnom bazom podataka koja ne sadrži indekse ni statistike, dok će se drugo mjerenje obaviti nad identičnom bazom podataka koja će dodatno imati primjerene, odnosno preporučene indekse i statistike. Eksperiment će se izvesti na način da se pokrene JavaScript skripta koja će svakih dvije sekunde izvršavati unos i ažuriranje podataka, a svakih pet sekundi će se zvati skripta koja će izvršiti osam upita. Mjeriti će se vrijeme u milisekundama koje je potrebno da se svaki od upita izvrši te će se oni zapisati u tekstualnu datoteku. Nakon 50 zapisa skripta će se zaustaviti i izračunati će se prosječno vrijeme potrebno za izvršenje pojedinog upita.

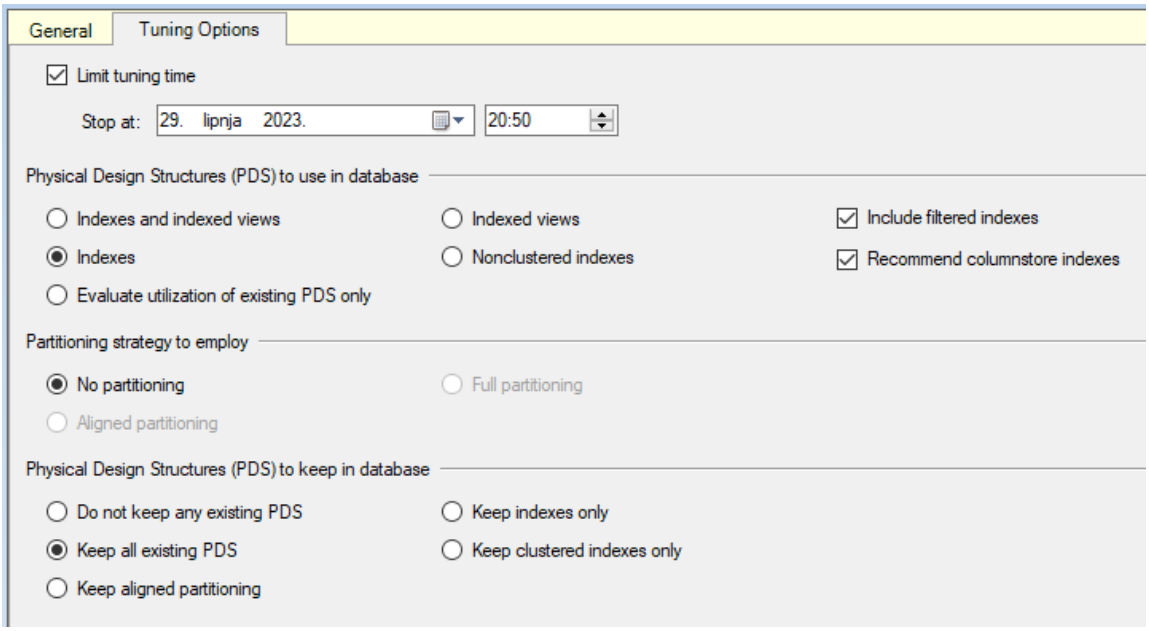
Proces razvoja aplikacije uključivao je primjenu nekoliko tehnologija. XAMPP, programski paket otvorenog koda, korišten je kao lokalno razvojno okruženje za olakšavanje postavljanja web poslužitelja. Ovo okruženje uključivalo je Apache kao web poslužitelj i Hypertext Preprocessor (PHP) kao skriptni jezik na strani poslužitelja. Microsoft SQL Server, robustan i skalabilan sustav upravljanja relacijskom bazom podataka, integriran je u arhitekturu aplikacije za učinkovito pohranjivanje i upravljanje podacima. Prednji dio aplikacije razvijen je pomoću HyperText Markup Language (HTML)-a, standardnog označnog jezika za izradu web stranica. HTML je pružio strukturni okvir i mogućnosti prezentacije sadržaja potrebne za izgradnju korisničkog sučelja. Uz HTML, korišten je i Asynchronous JavaScript and XML (AJAX) jQuery

i JavaScript kako bi se omogućila asinkrona komunikacija između klijentske i poslužiteljske strane. PHP, skriptni jezik na strani poslužitelja, odigrao je ključnu ulogu u procesu razvoja. Kombinirajući PHP s AJAX jQuery i JavaScriptom, ostvareno je dinamičko generiranje web stranica i interakcija s bazom podataka. Korištenjem AJAX jQuery, omogućeno je ažuriranje sadržaja web stranice bez potrebe za ponovnim učitavanjem cijele stranice. Za potrebe kodiranja, otklanjanja pogrešaka i upravljanja projektom korišten je Visual Studio Code, integrirano razvojno okruženje (Integrated Development Environment (IDE)) za izvorni kod. Visual Studio Code pružio je napredne alate i funkcionalnosti koje su olakšale razvoj aplikacije.

6.1 Optimizacija baze podataka

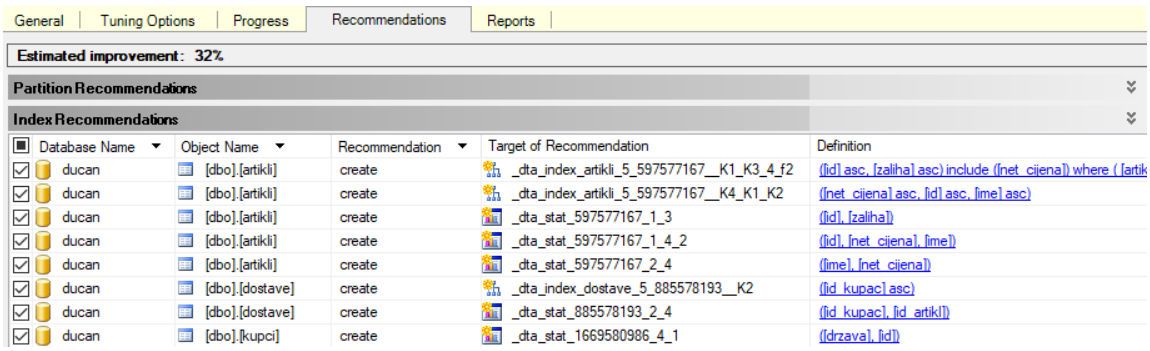
U svrhu dublje analize alata za optimizaciju i profiliranje baze podataka u Microsoft SQL Serveru, u eksperimentu će se dodatno uspoređivati rezultati dviju optimiziranih baza podataka - A i B.

Poboljšana baza podataka A predstavljat će rezultat optimizacije inicijalne baze koristeći radno opterećenje od jednog upita, točnije zasebno će se gledati dane preporuke za svaki pojedinačni upit. Osim toga koristit će se i savjeti procijenjenog plana izvršavanja koji također radi na principu analize jednog upita. Optimizator upita je pri analizi predložio sljedeće negrupirane indekse: stupac `drzava` u tablici `kupci`, stupac `net_cijena` u tablici `artikli` koji u sebi ima stupac `ime`, stupac `zaliha` u tablici `artikli` koji u sebi ima stupac `net_cijena`, stupac `drzava` u tablici `kupci` te stupac `id_artikl` u tablici `dostave` koji u sebi ima stupac `id_kupac`. Sa strane Microsoft savjetnika za optimizaciju baze podataka uvedene su statistike za zalihe, net cijenu, ime i id na artiklima, te države i id na kupcima. Osim toga, uvedena su i dva negrupirana indeksa koji koriste pohranu podataka temeljenu na stupcima i obradu upita (eng. *columnstore*) u tablici `dostave` i tablici `kupci`. Tijekom stvaranja dotičnih osam sjednica za analizu upita označena je postavka kojom se može očuvati postojeća fizička struktura baze (eng. *Physical Design Structures*). To znači da savjetnik ne bi preporučivao brisanje postojećih indeksa i statistika u sustavu. Sve opcije podešavanja koje su se koristile tijekom eksperimenta su predstavljene na slici 6.1.



Slika 6.1 *Opcije podešavanja sjednice u Microsoft savjetniku za optimizaciju baze podataka korištene u eksperimentu*

Poboljšana baza podataka B predstavljat će rezultat optimizacije koristeći radno opterećenje od skupa upita, što znači da će Microsoft savjetnik za optimizaciju baze podataka u jednoj sjednici analizirati svih osam upita odjednom. Na slici 6.2 prikazani su svi savjeti koji će u konačnici biti implementirani u sustav. Tijekom stvaranja sjednice koristile su se iste opcije podešavanja kao na slici 6.1, osim zadnje postavke gdje je omogućeno brisanje postojećih fizičkih struktura u bazi.



Slika 6.2 *Savjeti Microsoft savjetnika za optimizaciju baze podataka na temelju skupa od osam upita*

6.2 Mjerenje vremena odziva

Napravljena je JavaScript skripta koja pri pokretanju započinje odbrojavanje. Svake dvije sekunde se aktiviraju skripte za unos kupaca, artikala, dostavljača, vozila i dostava, a svakih pet sekundi se aktivira skripta koja će izvršiti osam specifičnih upita. Nad tim upitima će se mjeriti vrijeme odziva te će se rezultat zapisati u tekstualnu datoteku, pomoću koje će se u konačnici izračunati prosjek izvršavanja pojedinih upita. Nakon što se tih osam upita izvrše točno pedeset puta, skripta se zaustavlja te se prosjek zapisuje u posebnu tekstualnu datoteku. Implementacija navedenog procesa prikazana je u ispisu 6.1.

Isto mjerenje izvršit će se nad inicijalnom bazom, te identičnom bazom koja usput ima implementirane preporuke od savjetnika za optimizaciju baze podataka. Kako bi se osiguralo da su sve tri baze podataka u istom stanju prije pokretanja skripte, napravljena je sigurnosna kopija (eng. *backup*) inicijalne baze. Nakon mjerenja koristilo bi se sučelje MSS-a kako bi se vratila (eng. *restore*) sigurnosna kopija u trenutno stanje sustava. Time obje baze imaju jednake podatke i broj informacija u početku vođenja mjerenja.

```
$executionTimes = [];
if(file_exists('mjerenje.txt')){
    $executionTimes = unserialize(file_get_contents('mjerenje.txt'))
}
$executionTimes[] = $queryExecutions;
file_put_contents('mjerenje.txt',
serialize($executionTimes));
if($triggerCount == 49){
    $averageExecutionTimes = [];
    for($i = 0; $i < count($executionTimes[0]); $i++){
        $queryTimes = array_column($executionTimes, $i);
        $averageExecutionTimes[] = array_sum($queryTimes)/50;
    }
    file_put_contents('rezultat.txt',
    serialize($averageExecutionTimes));
    exit();
}
$triggerCount++;
file_put_contents('trigger_count.txt',
serialize([$triggerCount]));
```

Ispis 6.1 *Dio koda koji zapisuje vremena izvršavanja upita u tekstualnu datoteku, te pri 50. mjerenju izračunava prosjek*

6.3 Eksperimentalni upiti

U ovom se poglavlju navode i objašnjavaju svi upiti koji su se izvršavali i čije se vrijeme mjerilo tijekom eksperimenta.

- Prvi upit
  Dohvaćaju se tri najčešće naručena artikla time da se računa broj pojavljivanja svakog primarnog ključa artikla u tablici `dostave` te se rezultat sortira po broju pojavljivanja u silaznom redoslijedu. U nastavku se nalazi opisani SQL upit:

```
SELECT TOP 3 id_artikl, COUNT(*) AS artikl_count FROM dostave GROUP
   BY id_artikl ORDER BY artikl_count DESC
```

- Drugi upit

  Izračunava se prosjek neuspješnih dostava nad ukupnim brojem dostava za svaku pojedinu zemlju u tablici `drzave`. SQL upit ima sljedeći izgled:

```
SELECT countries.ime,
COUNT(dostave.id) AS dostave_1,
COUNT(neuspjesne_kupnje.id_artikl) AS dostave_2,
(COUNT(dostave.id)/COUNT(neuspjesne_kupnje.id_artikl))
* 100 AS postotak_dostava
FROM( SELECT drzave.ime, kupci.id FROM drzave
    INNER JOIN kupci ON kupci.drzava = drzave.ime
) AS countries
LEFT JOIN
dostave ON dostave.id_kupac = countries.id
LEFT JOIN
neuspjesne_kupnje ON neuspjesne_kupnje.id_kupac = countries.id
GROUP BY countries.ime
```

- Treći upit

  Vraća se prosječna količina artikla koja se naručuje u dostavama. Odgovarajući SQL upit izgleda ovako:

```
SELECT id_artikl, AVG(kolicina) AS avg_kolicina FROM dostave GROUP
   BY id_artikl
```

- Četvrti upit

  Dohvaća informacije o jednom artiklu s najvećom neto cijenom (vrijednost u stupcu `net_cijena`) te državi u kojoj je taj artikl najveći broj puta dostavljen. SQL upit ima sljedeći izgled:

```
SELECT TOP 1
artikli.ime AS artikl_ime,
kupci.drzava AS drzava_dostave
FROM artikli
INNER JOIN dostave ON dostave.id_artikl = artikli.id
INNER JOIN kupci ON kupci.id = dostave.id_kupac
WHERE
artikli.net_cijena = (SELECT MAX(net_cijena) FROM artikli)
GROUP BY artikli.ime, kupci.drzava
ORDER BY COUNT(*) DESC
```

- Peti upit

  Dohvaća se država s najvećim brojem dostava. SQL upit je prikazan u nastavku:

```
SELECT TOP 1 kupci.drzava,
COUNT(*) AS dostave_count
FROM dostave
INNER JOIN kupci ON dostave.id_kupac = kupci.id
GROUP BY kupci.drzava
ORDER BY COUNT(*) DESC
```

- Šesti upit

  Izračunava se prosječna udaljenost dostave za svaku dostavu, uz uvjet da je neto cijena artikla na dostavi između 250 i 10000. Opisani SQL upit izgleda ovako:

```
SELECT dostave.id, AVG(kupci.adresa) AS avg_distanca
FROM dostave
INNER JOIN artikli ON dostave.id_artikl = artikli.id
INNER JOIN kupci ON dostave.id_kupac = kupci.id
WHERE artikli.net_cijena BETWEEN 250 AND 10000
GROUP BY dostave.id
```

- Sedmi upit

  Za one artikle čiji primarni ključ nije prisutan u tablici `dostave`, odnosno artikli koji nisu nikada naručeni, te artikli čija zaliha iznosi između 11 i 299, prepolovi se neto cijena. SQL upit ima sljedeći izgled:

```
UPDATE artikli SET net_cijena = CAST(net_cijena * 0.5 AS INT) WHERE
    NOT EXISTS (
     SELECT 1 FROM dostave
     WHERE dostave.id = artikli.id
) AND zaliha BETWEEN 11 AND 299
```

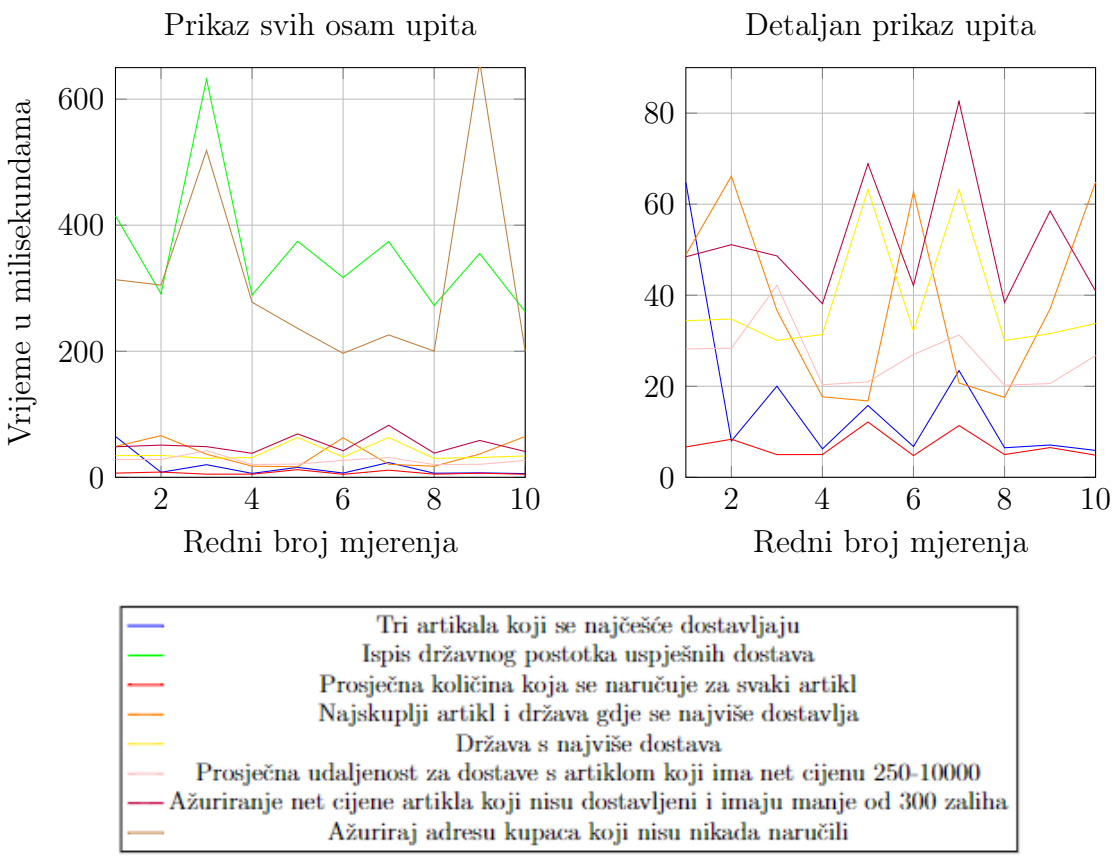
- Osmi upit

  Ažurira se adresa kupaca koji nisu nikada naručili, odnosno čiji se primarni ključ ne pojavljuje u tablici `dostave` u stupcu `id_kupac`. U nastavku se nalazi opisani SQL upit:

```
UPDATE kupci
SET adresa = CAST(adresa - (adresa / 1.6) AS INT)
WHERE id NOT IN (SELECT id_kupac FROM dostave);
```

6.4 Rezultati s inicijalnom bazom podataka

Na slici 6.3 prezentirano je deset (svaka peta) od pedeset vrijednosti dobivene tijekom mjerenja vremena izvršavanja pojedinih upita. Jasno je da su drugi i osmi upiti (obojani zelenom i smeđom bojom na grafu) u prosjeku trajali puno duže od ostalih upita. Radi bolje preglednosti su ostali upiti sa prosječnim vremenom izvršavanja ispod 80ms izdvojeni u zaseban graf.

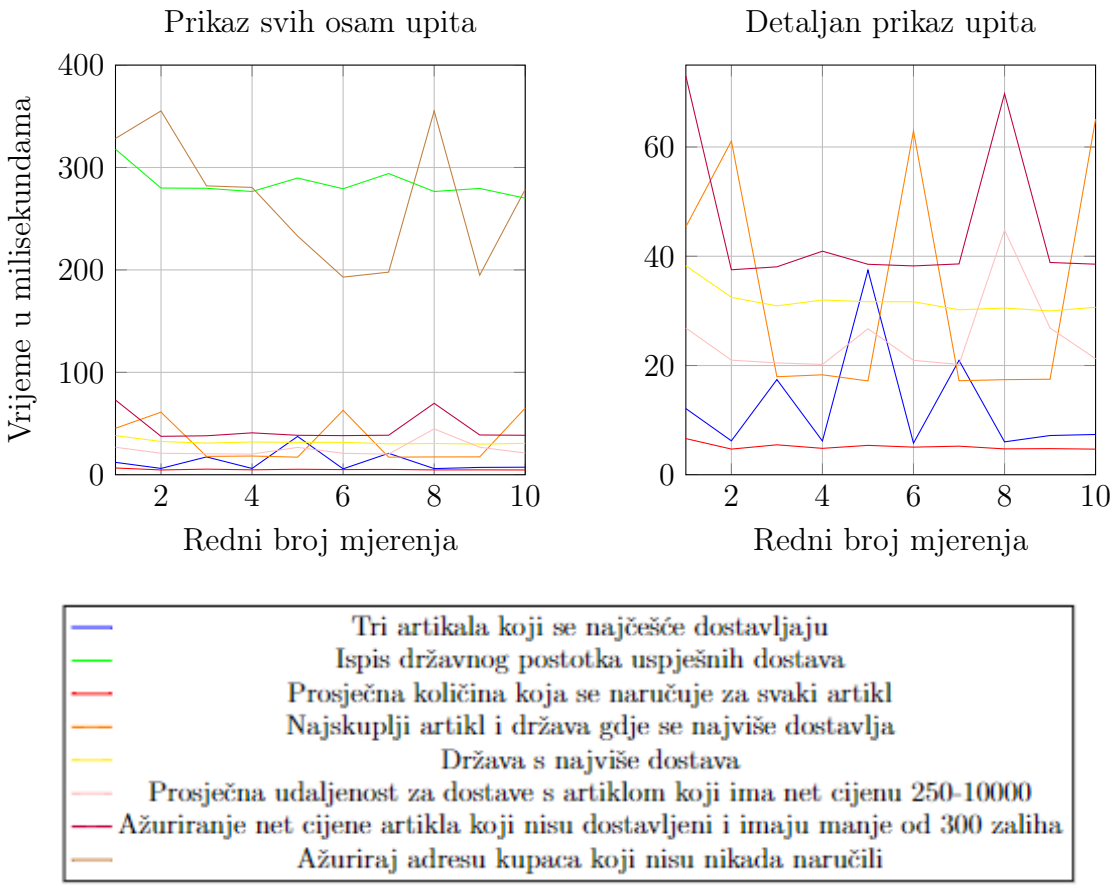


Slika 6.3 *Vrijednosti vremena odziva upita u inicijalnoj bazi podataka*

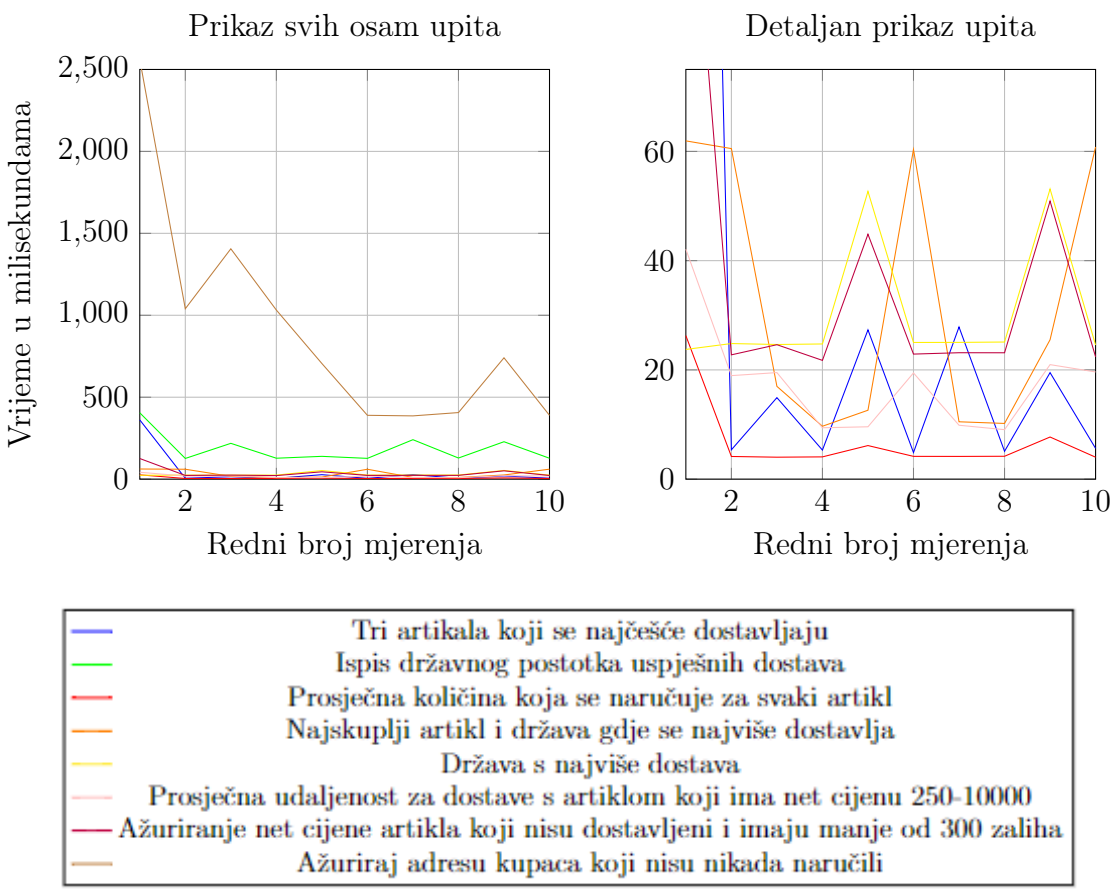
6.5 Rezultati s poboljšanom bazom podataka

Na slici 6.4 ilustrirani su grafovi koji prikazuju vrijeme izvršavanja upita u poboljšanoj bazi podataka A koja je u sebi imala savjete optimizatora upita i Microsoft savjetnika za optimizaciju baze podataka na temelju pojedinačnih upita.

Na slici 6.5 predstavljeni su grafovi koji prikazuju vrijeme izvršavanja upita u poboljšanoj bazi podataka B koja je u sebi imala samo savjete za poboljšanje od strane Microsoft savjetnika za optimizaciju baze podataka na temelju cijelog skupa od osam upita.



Slika 6.4 *Vrijednosti vremena odziva upita u poboljšanoj bazi podataka A*



Slika 6.5 *Vrijednosti vremena odziva upita u poboljšanoj bazi podataka B*

6.6 Analiza rezultata mjerenja

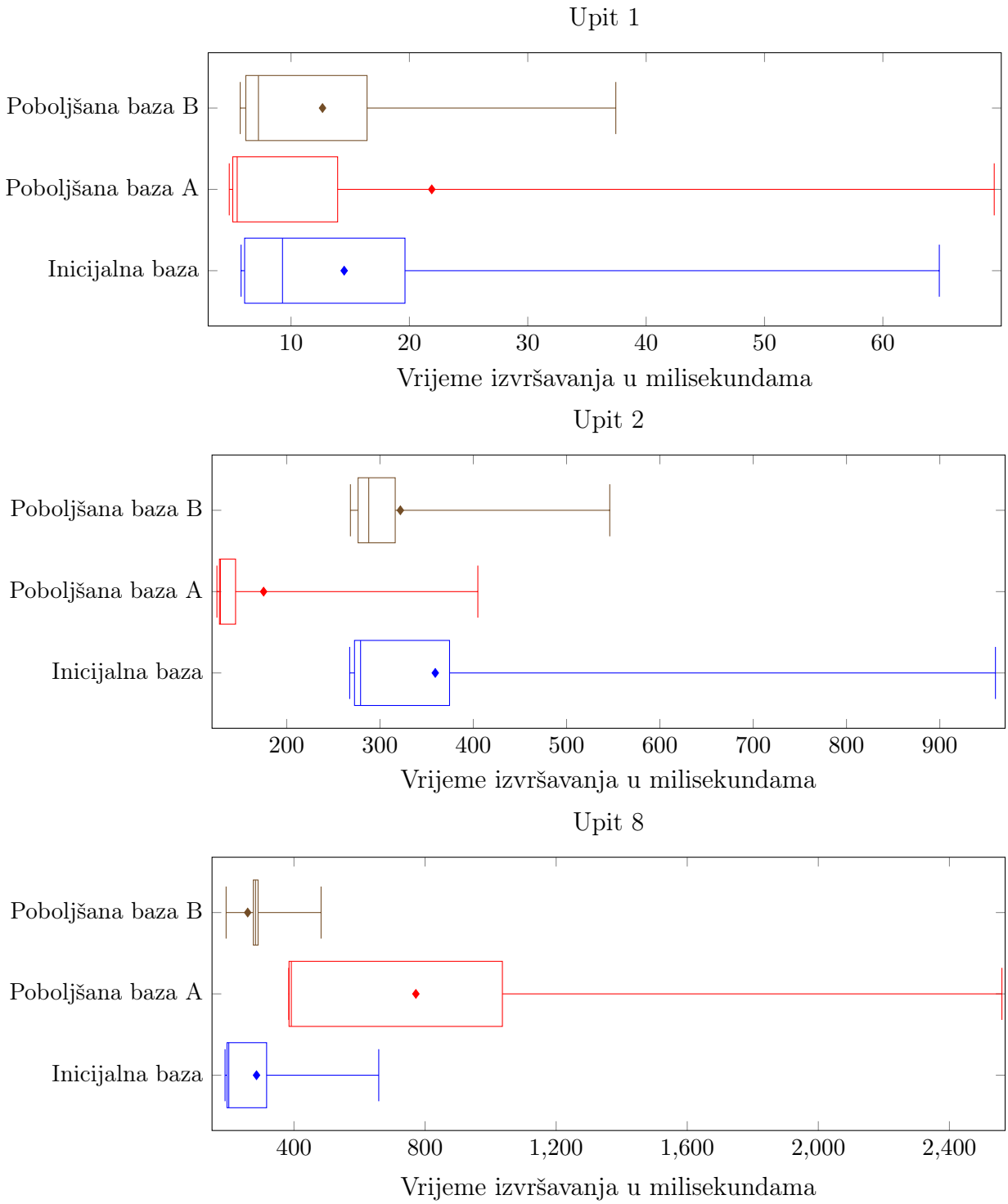
Izračunom sveukupnog prosjeka vremena izvršavanja pojedinih upita dobije se bolji uvid u razliku između tri baze podataka. U tablici 6.1 prikazana su prosječna vremena izvršavanja upita u milisekundama kod pojedinih baza podataka. Gledajući bazu podataka koja se optimizirala analizom pojedinačnih upita, šest od osam upita je optimizirano na način da im je prosječno vrijeme izvršavanja smanjeno. Oni prosjeci koji su se, suprotno očekivanju, povećali nakon uvođenja indeksa i statistika su podebljani u tablici.

Tablica 6.1 *Usporedba prosječnih vremena izvršavanja upita u različitim bazama podataka*

Upit	*Inicijalna baza podataka*	*Poboljšana baza podataka A*	*Poboljšana baza podataka B*
1	14.49	**21.89**	12.66
2	359.27	175.29	321.87
3	6.78	5.17	5.82
4	34.09	28.12	31.32
5	39.12	29.99	34.35
6	26.54	15.97	25.20
7	51.69	30.19	44.40
8	285.76	**771.99**	258.84

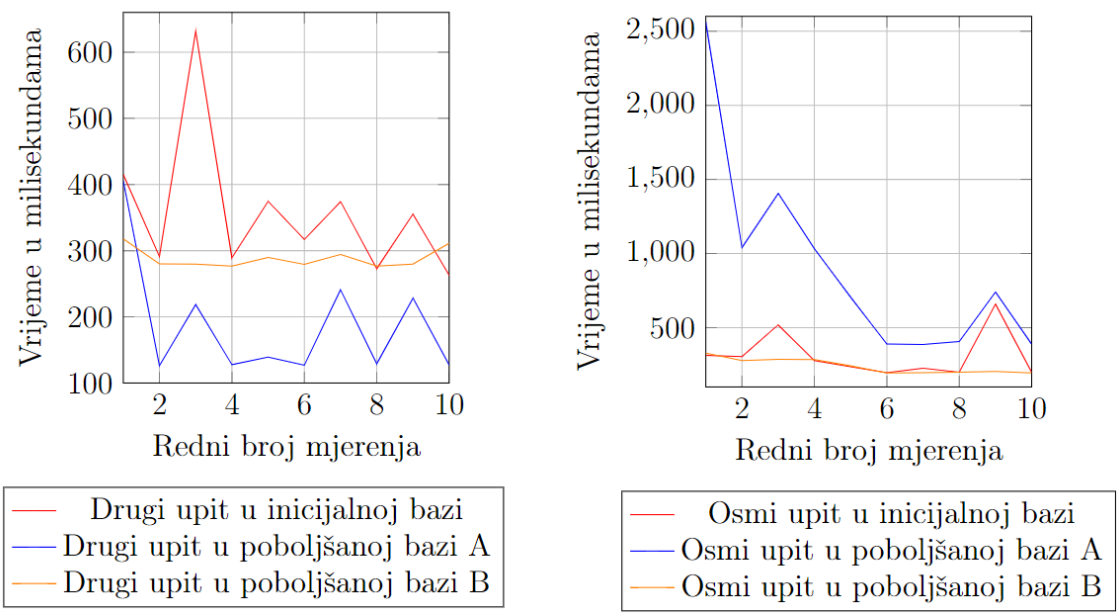
U analizi rezultata istaknut će se i pomnije istražiti tri od osam upita. Prvi i osmi upit izabrani su iz razloga što se njihovo prosječno vrijeme izvršavanja upita povećalo u poboljšanoj bazi podataka A. Drugi upit će se analizirat jer se njegovo vrijeme izvršavanja skoro prepolovilo u bazi A, a neznatno smanjilo u bazi B.

Na slici 6.6 predstavljeni su prosjek, medijan, te donja (eng. *minimum*) i gornja (eng. *maximum*) granica za istaknuta tri upita. Kod poboljšane baze na temelju većeg radnog opterećenja, osim samog prosjeka i medijana, varijacije (ili granice vrijednosti) su se također smanjile. Uspoređujući grafove inicijalne baze i poboljšane baze A može se vidjeti da nakon uvođenja preporuka savjetnika za optimizaciju donje granice su se uglavnom smanjile, ali se i većina gornjih granica povećala. Iako se kod nekih upita u poboljšanoj bazi A donja granica smanjila više nego kod poboljšane baze B, u potonjoj bazi nema upita kojima se vrijeme odziva povećalo u usporedbi s inicijalnom bazom.



Slika 6.6 *Deskriptivna statistika za vrijeme izvršavanja tri specifična upita u promatranim bazama podataka*

Na slici 6.7 ilustrirana su dva grafa za drugi i osmi upit gdje se može vidjeti njihovo vrijeme izvršavanja kroz sve tri baze podataka u eksperimentu. Prosječno vrijeme odziva drugog upita se u poboljšanoj bazi A smanjilo za 51.81%, a u poboljšanoj bazi B za 10.31%. Prosjek vremena odziva osmog upita se u poboljšanoj bazi A povećalo za 170.53%, a u poboljšanoj bazi B se smanjilo za 9.12%. Potencijalni uzrok usporavanja izvršavanja ovog upita u poboljšanoj bazi A je taj što postoji veći broj negrupiranih indeksa koji u sebi sadrže stupac `adresa`. Kao što je prethodno spomenuto, negrupirani indeksi mogu negativno utjecati na naredbe ažuriranja podataka zbog potrebe da se željeni podaci mijenjaju na više mjesta - na tablici i samom negrupiranom indeksu. Kako se već radi o velikom broju (102999) redaka koji se dohvaćaju, dodatno mijenjanje tog broja podataka značajno usporava upit. Iako su ti indeksi pomogli ubrzati druge upite, kao što je izračun državnih postotaka uspješnih dostava, uvelike su odmogli pri poboljšanju naredbi ažuriranja većeg broja podataka na tim tablicama. Ova pojava dokazuje da je korisnije raditi na analizama velikih radnih opterećenja gdje će savjetnik za optimizaciju baze podataka biti u mogućnosti donijeti globalno korisne savjete.



Slika 6.7 *Usporedba vremena izvršavanja dva upita prije i poslije uvođenja preporuka za optimizaciju baze podataka*

Poglavlje 7

Zaključak

U ovom završnom radu, analiza ostvarenih rezultata jasno ukazuje na učinkovitost i važnost alata za optimizaciju unutar Microsoft SQL upravitelja baze podataka. Uvođenjem složene strukture baze podataka koja simulira rad dućana, sa velikim brojem tablica i okidača, postignut je učinak sporog odziva sa strane poslužitelja. Gotovo sva funkcionalnost i logika dućana smještena je u same okidače baze, tako da svaki upit koji dodaje, mijenja ili briše podatke u tablicama ima visoki operativni trošak. Microsoft savjetnik za optimizaciju baze podataka uspješno je detektirao problematične točke u sustavu te dao korisne savjete koji bi pomogli optimizirati sustav. Navedeni savjeti preporučali su izgraditi statistike i negrupirane indekse nad određenim stupcima koji su se često pretraživali. Isto tako je i optimizator upita predstavio nekoliko savjeta pri izradi procijenjenog plana izvršavanja upita koji je pomogao smanjiti vrijeme odziva upita.

No, tijekom analiza i implementiranja savjeta pojedinačnih upita uočena je česta proturječnost ili nekonzistentnost kada bi se savjetniku predalo radno opterećenje koje se sastojalo od samo jednog upita. Razlog tome je što Microsoft savjetnik za optimizaciju baze podataka nastoji čim više optimizirati dobiveni upit, ne uzimajući u obzir kako bi njegovi savjeti utjecali na druge moguće upite i rubne slučajeve. Iako je krajnji rezultat učinkovit, može imati negativne posljedice na druge upite i naredbe. Indeksi predstavljaju pogodna rješenja za poboljšanje izvođenja, ali oni nisu pogodni u svim slučajevima. Oni omogućavaju brzo pretraživanje, filtriranje i sortiranje podataka temeljem određenih stupaca, ali zauzimaju dodatni prostor

na disku i zahtijevaju ažuriranje prilikom izmjene podataka, što može utjecati na izvršenje operacija umetanja, ažuriranja i brisanja.

Izuzetno je važno prvo provesti temeljitu analizu strukture baze, dobro poznavati izgled podataka u tablicama i najčešće naredbe koje se koriste nad podacima. Tako bi se planiranjem indeksiranja izbjegli nepotrebni indeksi i minimizirao negativan utjecaj indeksa na rad sustava. Na temelju istraživanja i praktičnih primjera, zaključeno je da su alati za optimizaciju u Microsoft SQL bazi podataka vrlo korisni i napredni alati za postizanje visoko-učinkovitih upita i brzih odziva. Preporuča se predati Microsoft savjetniku za optimizaciju baze podataka što veće radno okruženje kako bi njegove preporuke mogle imati globalno pozitivan utjecaj na rad sustava. Pravilno planiranje, dizajniranje i održavanje indeksa od vitalnog su značaja za osiguravanje optimalne izvedbe sustava i poboljšanje korisničkog iskustva. Kroz kritičku analizu dobivenih savjeta od strane MSS alata može se ostvariti veća produktivnost i smanjenje opterećenja na bazu podataka.

Bibliografija

[1] M. E. M. George Edward Pelham Box, *A Note on the Generation of Random Normal Deviates*, 2nd ed. O'Reilly, 1958.

[2] *Start and use the Database Engine Tuning Advisor*, Microsoft. , s Interneta, https://learn.microsoft.com/en-us/sql/relational-databases/performance/start-and-use-the-database-engine-tuning-advisor?view=sql-server-ver16

[3] E. Erkec, *Explore the secrets of SQL Server execution plans*, SQLShack. , s Interneta, https://www.sqlshack.com/explore-the-secrets-of-sql-server-execution-plans/

[4] R. Gupta, *SQL Server execution plan — what is it and how does it help with performance problems?*, Quest. , s Interneta, https://blog.quest.com/sql-server-execution-plan-what-is-it-and-how-does-it-help-with-performance-problems/

[5] S. Yambadwar, *Difference between Clustered and Non-clustered index*, GeeksForGeeks. , s Interneta, https://www.geeksforgeeks.org/difference-between-clustered-and-non-clustered-index/

[6] B. Richardson, *What is the difference between Clustered and Non-Clustered Indexes in SQL Server?*, SQLShack. , s Interneta, https://www.sqlshack.com/what-is-the-difference-between-clustered-and-non-clustered-indexes-in-sql-server/

[7] *Clustered and nonclustered indexes described*, Microsoft. , s Interneta, https://learn.microsoft.com/en-us/sql/relational-databases/indexes/clustered-and-nonclustered-indexes-described?view=sql-server-ver16

[8] N. Bharti, *Advantages and Disadvantages of Indexing in SQL*, Scaler. , s Interneta, https://www.scaler.com/topics/sql/advantages-and-disadvantages-of-indexing-in-sql/

[9] *Statistics*, Microsoft. , s Interneta, https://learn.microsoft.com/en-us/sql/relational-databases/statistics/statistics?view=sql-server-ver16

Pojmovnik

AJAX Asynchronous JavaScript and XML. 24

CPU Central Processing Unit. 14

HTML HyperText Markup Language. 24

IDE Integrated Development Environment. 25

META Microsoft Engine Tuning Advisor. 9, 13

MSS Microsoft SQL Server. 1, 2, 9, 10, 14, 21, 23, 27, 38

PHP Hypertext Preprocessor. 24, 25

SQL Structured Query Language. 6, 9, 10, 21, 23

Sažetak

U ovom radu analiziran je proces i učinak profiliranja i optimizacije baze podataka u sustavu SQL Server. Razvijen je ogledni sustav s potpornom bazom podataka u koju je uveden veliki broj inicijalnih zapisa i veći broj okidača. Na taj je način simulirano odgovarajuće opterećenje na sustav za upravljanje bazama podataka koje rezultira adekvatnim povećanjem vremena odziva na upite. Primjenom alata za automatsko profiliranje i optimizaciju unutar sustava SQL Server, postignuta su odgovarajuća poboljšanja performansi. Konkretno, korištenjem preporuka koje se zasnivaju na planu izvršavanja upita i procijenjenom trošku, provedene su akcije poput stvaranja negrupiranih indeksa i statistika nad odgovarajućim atributima u tablicama baze podataka. Detaljnom analizom eksperimentalnih rezultata dobiven je bolji uvid u učinak automatskog profiliranja i optimizacije. Pokazano je da prijedlozi za optimizaciju nisu uspješni u svakom scenariju te da je važno oprezno primjenjivati predložene indekse i statistike. Također, utvrđeno je da alatu za optimizaciju treba predati čim veće radno opterećenje kako bi generirane preporuke imale globalno pozitivan utjecaj na rad sustava.

Ključne riječi **— baze podataka, profiliranje, optimizacija, SQL Server**

Abstract

This thesis analyzes the process and effect of database profiling and optimization in SQL Server. An example system was developed with a supporting database in which a large number of initial records and a large number of triggers were introduced. In this way, the corresponding load on the database management system was simulated, resulting in a respective increase in the response time to queries. By using tools for automatic profiling and optimization within SQL Server, corresponding performance improvements were achieved. In particular, recommendations based on the query execution plan and estimated costs were used to implement measures such as the creation of non- clustered indexes and statistics on the corresponding attributes in the database tables. A detailed analysis of the experimental results provided a better insight into the effect of automatic profiling and optimization. It is shown

that the optimization suggestions are not successful in every scenario and that it is important to apply the suggested indexes and statistics judiciously. It was also found that the tuning advisor should be given a larger workload so that the generated recommendations have a global positive impact on the operation of the system.

Keywords **— databases, profiling, optimization, SQL Server**