Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Dušan Berce

Spremljanje izvajanja poizvedb na podatkovnem strežniku

DIPLOMSKA NALOGA
NA UNIVERZITETNEM ŠTUDIJU

doc. dr. Iztok Lebar Bajec
MENTOR

Ljubljana, 2010

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01659/2010
Datum: 05.04.2010

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **DUŠAN BERCE**

Naslov: **SPREMLJANJE IZVAJANJA POIZVEDB NA PODATKOVNEM STREŽNIKU**

MONITORING QUERY EXECUTION ON A DATABASE SERVER

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

V sodobni, k informacijam usmerjeni, družbi postajajo informacijske tehnologije vse pomembnejši del našega vsakdana. S tem pa postajata vse pomembnejši tako zanesljivost, kot tudi zmogljivost delovanja informacijske tehnologije. V poslovnem svetu je najpomembnejši člen informacijski sistem, ki nudi podporo poslovnemu procesu. Srce informacijskega sistema, s stališča zmogljivosti njegovega delovanja, predstavlja podatkovni strežnik.

V diplomski nalogi preučite obstoječe metode spremljanja izvajanja poizvedb na podatkovnem strežniku. Na primeru namišljenega podjetja prikažite metodologijo spremljanja izvajanja poizvedb tako s stališča skrbnikov informacijskega sistema kot tudi razvijalcev programske opreme. Prikažite tudi postopek odločanja o korakih, ki bodo zagotovili izboljšanje izvajanja opazovanih poizvedb.

Mentor:

doc. dr. Iztok Lebar Bajec

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:

prof. dr. Franc Solina

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Dušan Berce

Spremljanje izvajanja poizvedb na podatkovnem strežniku

POVZETEK

Informatika in z njo povezana analitika sta v sodobnem svetu postali nepogrešljiv del poslovnih procesov družb, organizacij in podjetij. Zanesljivost in točnost relevantnih poslovnih podatkov zagotavljata v zahtevnem okolju konstantno gospodarsko rast in večjo preglednost poslovanja. Zaradi tega dajejo poslovni sistemi velik poudarek celovitim poslovno analitičnim rešitvam, ki jim omogočajo večjo konkurenčno prednost.

Za kakovostno opravljanje strežniških storitev je ključnega pomena optimalnost strežniških odzivnih časov ter prepustnost servisiranja, kar se odraža v zasedenosti sistemskih virov. S ciljem ohranjanja kakovosti opravljanja strežniških storitev se pojavi potreba po orodjih in postopkih, ki omogočajo razvijalcem in skrbnikom poslovnih poročil spremljanje izvajanja opravil na strežniku.

Pričujoča diplomska naloga se na primeru namišljenega oglaševalskega podjetja ukvarja z analizo zmogljivosti delovanja podatkovnega strežnika in predstavlja možen pristop k reševanju zmogljivostnih težav le-tega. S pomočjo v diplomi opisanih orodij in postopkov sem se osredotočil na izvajanje poizvedb, ki so generirala poslovna poročila.

Ključne besede: zmogljivost podatkovnega strežnika, spremljanje izvajanja poizvedb, poslovna poročila

University of Ljubljana
Faculty of Computer and Information Science

Dušan Berce

Monitoring query execution on a database server

ABSTRACT

Information technology and analytics have become an indespensable part of business processes in companies and organizations. Reliability and accuracy of relevant data ensure constant economic growth and better business operations in highly demanding environment. That is why business systems put a lot of emphasis to the complete analytical solutions which assure them better competetive edge.

Optimum data base server response time and permeability of servising are significant for quality data base server performance. This reflects in occupation of system sources. With the goal of preserving quality of data base server performance originates the need to use tools and procedures that give developers and administrators of business reports chance to perform services on the server.

This diploma centers around database server performance analysis in imaginary advertising company and presents possible approach to solving its performance problems. I my diploma I give focus on performing inquiries with the help of specific tools and procedures which in return generated business reports.

Key words: Database server performance, Monitoring query execution, Buisness reports

ZAHVALA

Zahvaljujem se mentorju doc. dr. Iztoku Lebarju Bajcu za strokovne usmeritve in nasvete, prav tako tudi prof. dr. Nikolaju Zimicu za začetni zagon pri definiciji predmeta pisanja. Posebej bi se rad zahvalil svojemu stricu Janezu Bercetu za koristne komentarje k vsebini diplome. Nenazadnje bi se rad zahvalil tudi Luciji za pomoč pri končni obliki diplome in za dolgoletno moralno podporo.

— Dušan Berce, Ljubljana, junij 2010.

KAZALO

Povzetek i

Abstract iii

Zahvala v

1 Uvod 1

1.1 Opredelitev problema 1

1.2 Analiza zmogljivosti sistema 2

2 Opis sistema 5

2.1 Informacijski sistem podjetja 5

2.2 Podatkovni strežnik 6

2.3 Transakcijska podatkovna baza 7

2.4 Podatkovno skladišče 9

2.5 Polnjenje podatkovnega skladišča 12

3 Metode spremljanja izvajanja poizvedb 17

3.1 Odkrivanje zmogljivostnih težav 17

3.2 Zasedenost sistemskih virov 18

3.3 Sledenje izvajanja poizvedb 20

3.4 Načrt izvrševanja poizvedb 23

4 Izvedba spremljanja izvajanja poizvedb 27

4.1 Določitev testnih bremen 27

4.2 Vzpostavitev začetnega stanja 29

4.3 Meritve na testnih bremenih 30

4.3.1 Kazalnik KPI 31
4.3.2 Prihodki na projektih za tekoče in prejšnje leto 36
4.3.3 Rezultat po naročnikih med leti 2006 in 2010 40

5 Zaključek **47**
5.1 Sklepne ugotovitve 47

Literatura **49**

1 Uvod

1.1 Opredelitev problema

Informatika in z njo povezana analiza podatkov sta v sodobnem svetu postali nepogrešljiv del poslovnih procesov družb, organizacij in podjetij. Zanesljivost in točnost relevantnih podatkov zagotavljata v zahtevnem poslovnem okolju konstantno gospodarsko rast in večjo preglednost poslovanja. Zaradi tega dajejo poslovni sistemi velik poudarek celovitim analitičnim rešitvam, ki jim omogočajo večjo konkurenčno prednost.

Predpogoj za uspešno analizo poslovanja podjetja je, da se v njihovih informacijskih sistemih hrani čim večja količina podatkov, ki morajo biti v povezavi s poslovnimi procesi ustrezno strukturirani. Informacijsko podporo delovnim procesom običajno nudi podatkovni strežnik, ki predstavlja jedro informacijskega sistema podjetja. Uporabniki preko uporabniškega vmesnika vnašajo podatke v podatkovno bazo, pri čemer morata vmesnik in podatkovni strežnik skrbeti za integriteto podatkov. Kakovostni podatki namreč zagotavljajo večjo zanesljivost poslovnega poročanja.

Za kakovostno opravljanje strežniških storitev je ključnega pomena optimalnost strežniških odzivnih časov ter prepustnost servisiranja, kar se odraža v zasedenosti sistemskih

virov, hkrati pa se skozi življenjski cikel informacijskega sistema spreminjajo njegove lastnosti, kot so:

- število uporabnikov,
- količina podatkov,
- kompleksnost poizvedb,
- hitrost omrežnih povezav.

S ciljem ohranjanja kakovosti opravljanja strežniških storitev ne glede na spremembe naštetih lastnosti se pojavi potreba po orodjih in postopkih, ki omogočajo razvijalcem poslovnih poročil (v nadaljevanju razvijalcem) in skrbnikom podatkovnega strežnika (v nadaljevanju skrbnikom) spremljanje izvajanja opravil na podatkovnem strežniku[1]. Pričujoča diplomska naloga se ukvarja z analizo zmogljivosti delovanja podatkovnega strežnika in predstavlja možen pristop k reševanju zmogljivostnih težav le tega. V diplomski nalogi sem se osredotočil na spremljanje izvajanja poizvedb, ki generirajo poslovna poročila. Moj cilj je bil odgovoriti na sledeča vprašanja:

- Kako odkriti časovni interval, v katerem je podatkovni strežnik preobremenjen?
- Kako odkriti poizvedbo na podatkovnem strežniku, ki povzroči preobremenitev?
- Kako pojasniti razloge, zaradi katerih poizvedba povzroči preobremenitev?

1.2 Analiza zmogljivosti sistema

Konkretne korake pri načrtovanju vsebine diplomske naloge sem zastavil na sledeči način. Uporabil sem sistematični pristop predstavljen v knjigi [1], ki mi je dala dobra referenčna izhodišča za nadaljnje delo. Knjiga se ukvarja z osnovnimi pojmi računalniške zmogljivosti, kot so:

- pristopi k analizi,
- planiranje zmogljivosti z metodami in metrikami,
- obrazložitev pojma bremena,

[1] Razvijalci in skrbniki opravljajo ključno vlogo pri procesu poslovnega poročanja, kjer prvi skrbijo za izdelavo poročil, drugi pa za nemoteno delovanje informacijskega sistema.

- delovanje monitorjev,
- metode meritev, simulacije in analitični pristopi,
- predstavitev rezultatov analize.

Začetna koraka analize sta bila dosleden opis opazovanega sistema in postavitev ciljev same analize. S tem sem omogočil izbiro metodologije, ki daje uporabne rezultate. Opazovani sistem, ki ga predstavlja podatkovna baza oglaševalskega podjetja, sem opisal v drugem poglavju. Metodologijo analize sem podal v tretjem poglavju, kjer sem opisal orodja, ki omogočajo potrebno spremljanje izvajanja poizvedb na podatkovnem strežniku. Cilj opisa in prikaza uporabe orodij je:

- skrbnikom omogočiti odkrivanje poizvedb, ki povzročajo povečano obremenitev,
- razvijalcem omogočiti vpogled v zgradbo poizvedb.

V četrtem poglavju sem kot primer uporabe orodij in pristopov podal poizvedbe, ki generirajo poslovna poročila, in povzročajo preobremenitev podatkovnega strežnika. Vsako poizvedbo sem analiziral v dveh različicah:

- prva predstavlja izvorno različico algoritma, pri kateri so skrbniki odkrili povečano uporabo sistemskih virov,
- druga predstavlja poskus optimizacije algoritma s strani razvijalcev.

V istem poglavju sem podal tudi rezultate meritev izvajanja poizvedb, ki sem jih pridobil s pomočjo predstavljenih orodij. Pri tem sem opisal okoliščine, v katerih sem meritve opravljal. V poglavje sem vključil tudi interpretacijo rezultatov, kjer sem skušal odgovoriti na vprašanje, ali za pohitritev izdelave poslovnih poročil zadostuje optimizacija algoritmov, ali je morda potrebna nadgradnja strojne opreme.

V zaključku sem strnil in komentiral pridobljene rezultate ter podal možnost uporabe opisanega pristopa v realnem poslovnem okolju.

2 Opis sistema

2.1 Informacijski sistem podjetja

Informacijski sistem omogoča zbiranje, obdelavo, shranjevanje, distribucijo in uporabo informacij. Podjetjem nudi učinkovitejše upravljanje poslovnih procesov, s katerimi izvajajo svoje aktivnosti. Če je obseg poslovanja podjetja velik, je uporaba informacijskega sistema toliko bolj upravičena. Z velikostjo namreč postane upravljanje poslovnih procesov težje obvladljivo, zato se pojavi potreba po učinkoviti sistematizaciji in avtomatizaciji le-teh. To v praksi pomeni, da se v delovne procese podjetja uvede informacijsko tehnologijo.

V diplomski nalogi sem si zamislil podjetje, ki se ukvarja z oglaševanjem, in je hkrati dovolj veliko, da za učinkovito upravljanje svojih poslovnih procesov potrebuje informacijski sistem. Glavna aktivnost podjetja je izvajanje različnih trženjsko-komunikacijskih projektov za naročnike iz različnih družbenih sfer[1]. Trženjsko-komunikacijski projekt se vodi na sledeči način:

- določijo se osnovni podatki o projektu,

[1] Naročniki so lahko gospodarske družbe, vladne in nevladne organizacije, društva, itd.

- določi se projektni tim,
- formalizira se poslovno sodelovanje (pogodba),
- definira se finančni načrt,
- vodijo se finančne transakcije.

Med osnovne podatke o projektu spadajo ime projekta, naročnik, tip storitve in trajanje projekta. Projektni tim sestavljajo vodje projekta in ostali zaposleni, ki sodelujejo na njem. To so ponavadi oblikovalci, tekstopisci, trženjski in medijski analitiki, asistenti itd. Formalizacija poslovnega sodelovanja pomeni podpis pogodbe, ki je pravno zavezujoč dokument za naročnika in oglaševalsko podjetje. Finančni načrt podaja v času trajanja projekta oceno prihodkov in stroškov. Finančne transakcije na projektu se izvajajo preko prejetih in izdanih računov. Prejeti računi vsebujejo stroške potrošnega in promocijskega materiala (npr. nosilci podatkov, fotokopije, embalaža), zunanjih podizvajalcev (npr. tiskarji, prevajalci, lektorji), posrednikov (npr. agencije za medijski zakup, organizacije, ki urejajo avtorske pravice) in stroške pravnih storitev, ki nastanejo pri izvajanju projekta. Z izdanimi računi podjetje zaračuna naročniku vse stroške iz prejetih računov in svoje delo.

V informacijskem sistemu oglaševalskega podjetja se hranijo vsi zgoraj omenjeni podatki. Za sprejemanje strateških poslovnih odločitev potrebuje vodstvo podjetja poročila, iz katerih je razvidna uspešnost projektov. Hkrati pa ta poročila omogočajo tudi vpogled v trende poslovanja. Poglavitna lastnost poročil je, da dajejo relevantne povzetke iz velikih količin podatkov. Obdelava velikih količin podatkov tehnološko gledano pomeni povečano obremenitev sistemskih virov podatkovnega strežnika. V diplomskem delu sem se osredotočil na obravnavo preobremenitev sistemskih virov podatkovnega strežnika omenjenega podjetja, natančneje ozkih grl pri izvajanju poizvedb na podatkovnem strežniku.

2.2 Podatkovni strežnik

Podatkovni strežnik igra osrednjo vlogo v informacijskem sistemu podjetja, ker hrani podatkovno bazo in uporabnikom omogoča dostop do nje. Za realizacijo diplomske naloge sem uporabil strojno opremo, ki je navedena v tabeli 2.1, ter operacijski sistem Microsoft Windows XP SP3.

Parameter	Vrednost
CPE	Intel Core Duo 2.4 GHz
Velikost pomnilnika	3 GB
Trdi disk (kapaciteta)	300 GB
Trdi disk (hitrost vrtenja)	5400 rpm
Mrežna kartica	Marvell Yukon 88E8040T PCI-E Fast Ethernet

Tabela 2.1 Uporabljena strojna oprema pri diplomski nalogi.

Uporabil sem brezplačno verzijo Microsoftovega podatkovnega strežnik SQL Server 2005 Express Edition, ki ima v primerjavi s plačljivimi verzijami naslednje omejitve:

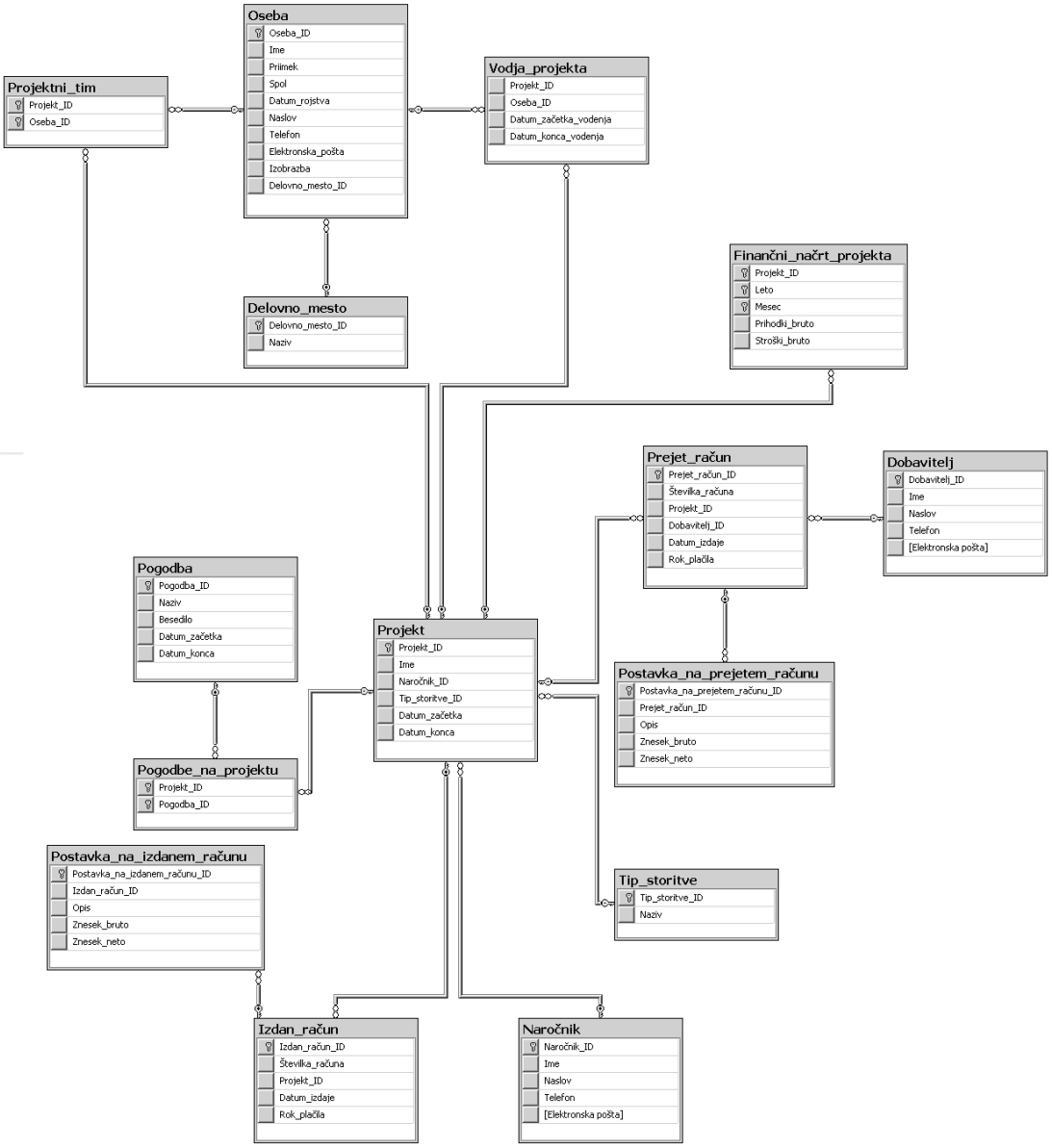
- največja velikost posamezne podatkovne baze je 4 GB,
- uporaba največ 1 GB glavnega pomnilnika,
- uporaba samo ene CPE.

Dodatno sem si namestil brezplačni grafični uporabniški vmesnik SQL Server Management Studio Express, ki ni del namestitvene distribucije. Ta omogoča enostavno upravljanje strežnika in poganjanje poizvedb v jezikih ANSI SQL in T-SQL. Vključuje tudi orodja za spremljanje dogajanja na strežniku, ki sem jih uporabil pri izvedbi meritev.

2.3 Transakcijska podatkovna baza

V nadaljevanju je opisan načrt podatkovne baze, ki jo podjetje uporablja za hranjenje informacij o poslovnih procesih[2] in je prikazan na sliki 2.1. Osnovni podatki o projektu so shranjeni v tabeli `Projekt`, razen informacije o vodji projekta, ki je shranjena v tabeli `Vodja_projekta`. Na ta način lahko pride do menjave vodje med samim izvajanjem projekta, trajanje vodenja pa opisujeta atributa `Datum_začetka_vodenja` in `Datum_konca_vodenja`. Podprta je tudi možnost, da v danem trenutku projekt vodi več oseb, vendar kontrola pri vnosu podatkov preko uporabniškega vmesnika to onemogoča. Tabela vsebuje še sklica na seznam naročnikov, ki so shranjeni v tabeli `Naročnik` ter na seznam tipov storitev, ki jih podjetje opravlja. Slednje je shranjeno v tabeli `Tip_storitve`.

[2]Poslovni procesi podjetja so opisani v podpoglavju 2.1.



Slika 2.1 Načrt transakcijske podatkovne baze oglaševalskega podjetja.

Informacija o projektnih timih je shranjena v tabeli `Projektni_tim`. Ta vodi evidenco zaposlenih oseb, ki so vključene v projekt. Seznam oseb je shranjen v tabeli `Osebe`, ki za vrednost atributa delovno mesto uporablja ločeno tabelo `Delovno_mesto`. Pogodbe so shranjene v tabeli `Pogodba`, vključitev na izbran projekt pa omogoča tabela `Pogodbe_na_projektu`. Tabela `Finančni_načrt_projekta` hrani načrtovane vrednosti prihodkov in stroškov na projektu za izbran mesec v izbranem letu.

Tabeli `Prejet_račun` in `Postavka_na_prejetem_računu` omogočata evidenco prejetih računov na projektih. Prva hrani podatek o številki računa, datumu izdaje, roku plačila ter vsebuje sklic na seznam dobaviteljev, ki so shranjeni v tabeli `Dobavitelj`. Druga pa hrani množico posameznih artiklov s pripadajočimi zneski. Podobno je organizirana tudi evidenca izdanih računov, kjer tabela `Izdan_račun` vodi osnovne informacije o računu, tabela `Postavka_na_izdanem_računu` pa hrani množico posameznih artiklov, ki jih podjetje zaračunava naročniku.

Predstavljeni načrt podatkovne baze, ki ga oglaševalsko podjetje uporablja za hranjenje informacij o poslovanju, je optimiziran za sprotno obdelavo transakcij (angl. OnLine Transaction Processing – OLTP). Transakcije dodajajo, brišejo in spreminjajo zapise v tabelah, k pospešitvi izvajanja teh operacij pa prispevata tudi normalizacija podatkov[3] in majhno število indeksov[4]. V OLTP sistemih je priporočljiva zmerna uporaba indeksov, ker je pohitritev prisotna predvsem pri poizvedbah, ki podatke berejo. Pri poizvedbah, ki podatke dodajajo, brišejo ali spreminjajo, pa popravljanje indeksnih struktur predstavlja dodatno obremenitev, ki se s količino podatkov in kompleksnostjo indeksov lahko močno poveča.

2.4 Podatkovno skladišče

Optimizacija načrta podatkovne baze za izvajanje transakcij, ki dodajajo, brišejo in spreminjajo podatke, ima za posledico upočasnitev poizvedb, ki podatke berejo. Od tod izvira motivacija po uporabi dodatnih podatkovnih objektov, ki so optimizirani za branje podatkov. Ti objekti, imenovani tudi podatkovno skladišče (angl. data warehouse), hranijo samo podatke, ki so potrebni za izdelavo poročil [2].

[3] Normalizacija podatkov je proces odpravljanja redundance v podatkih z namenom ohranjanja integritete ter odprave nepravilnosti pri njihovem urejanju.

[4] Indeks je podatkovna struktura, ki omogoča hitrejši dostop do podatkov (v primeru podatkovnega strežnika Microsoft SQL Server je to B-drevo).

Polnjenje skladišča s podatki iz transakcijske podatkovne baze [3] se realizira s procesom izluščevanja podatkov (angl. Extract Transform Load - ETL). Proces se običajno izvaja enkrat dnevno v času minimalne aktivnosti uporabe informacijskega sistema. Primerjava lastnosti transakcijske podatkovne baze in podatkovnega skladišča je prikazana v tabeli 2.2.

Transakcijska podatkovna baza	Podatkovno skladišče
- dodajanje, brisanje, spreminjanje in branje podatkov	- branje podatkov
- transakcije praviloma obdelujejo majhno količino podatkov	- poizvedbe lahko obdelujejo zelo velike količine podatkov
- načrt podatkovne baze je podrejen dobremu odzivu transakcij	- načrt podatkovne baze je podrejen dobremu odzivnemu času poizvedb
- podatki se polnijo preko uporabniškega vmesnika	- podatki se polnijo avtomatsko
- dinamično spreminjanje podatkov	- podatki so statični, le občasno ažurirani
- struktura poročil se redko spreminja	- strukturo poročil prilagajamo potrebam

Tabela 2.2 Primerjava lastnosti transakcijske podatkovne baze in podatkovnega skladišča.

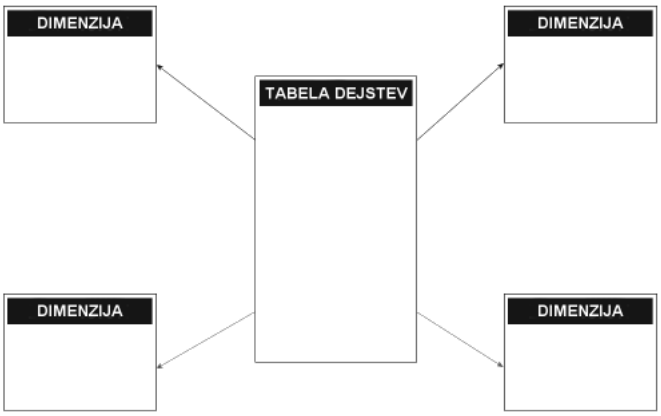
V diplomski nalogi sem podatkovno skladišče realiziral z zvezdno shemo (angl. Star Schema). Ta vsebuje osrednjo tabelo, ki se imenuje tabela dejstev in je obkrožena z večjim številom dimenzijskih tabel. Tabela dejstev hrani izključno numerične podatke dveh vrst:

- ključe do atributov v dimenzijskih tabelah, ki opisujejo dejstvo,
- numerične vrednosti meritev, ki pripadajo kombinaciji atributov.

Dimenzijske tabele hranijo atribute dejstev in običajno hranijo precej manjše število zapisov kot tabela dejstev. Primer zvezdne sheme je prikazan na sliki 2.2.

Pomembna lastnost tabele dejstev je zrnatost (angl. granularity), ki predstavlja frekvenco izluščevanja podatkov o dejstvih[5]. V diplomski nalogi sem za osnovno časovno

[5] Več na http://www.1keydata.com/datawarehousing/fact-table-granularity.html.



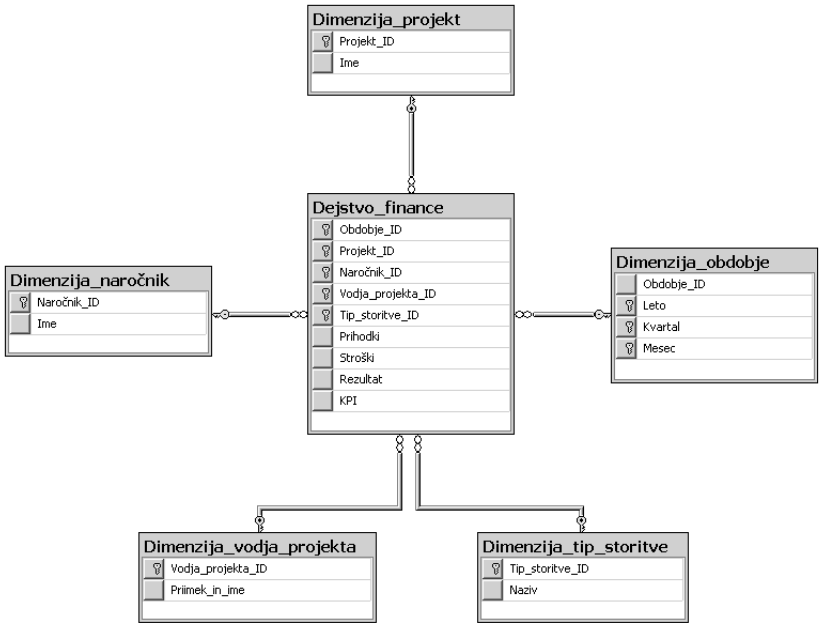
Slika 2.2 Zvezdna shema.

enoto uporabil mesec, z višanjem frekvence izluščevanja pa se lahko število zapisov v tabeli dejstev močno poveča, s čimer se upočasni izvajanje poizvedb. Izbor zrnatosti predstavlja kompromis med informativnostjo izpisov ter hitrostjo izvajanja poizvedb, ki se lahko zaradi vnaprej pripravljenih agregacij bistveno poveča.

Pospešitev poizvedb je pri zvezdni shemi navzoča tudi zaradi manjšega števila stikov (angl. join) med tabelami kot pri enakovrednih poizvedbah v transakcijski podatkovni bazi. Stik je razmeroma počasna operacija, a pri poizvedbah nad tabelo dejstev omogoča dostop do atributov v dimenzijskih tabelah. Za dostop do informacije o atributu dejstva se pri zvezdni shemi izvrši le ena operacija stika do dimenzijske tabele, ker te ne vsebujejo sklicev na druge tabele. V primeru, ko dimenzijske tabele vsebujejo sklice na druge tabele, zvezdna shema postane t.i. snežinkasta shema (angl. snowflake schema). K pospešitvi poizvedovanja pripomore tudi uporaba kompleksnejših indeksov ter selitev podatkovnega skladišča na drug strežnik, s čimer se doseže ločitev transakcijskega in poročevalskega servisiranja.

Podatkovno skladišče obravnavanega oglaševalskega podjetja hrani mesečne agregacije finančnih podatkov (prihodki, stroški, rezultat) ter kazalnik KPI (ključni performančni indikator) na projektih v tabeli dejstev z imenom `Dejstvo_finance`. V njej se poleg numeričnih podatkov hranijo še ključi do opisnih podatkov o naročniku, vodji in tipu storitve iz dimenzijskih tabel. Shema je prikazana na sliki 2.3.

Posebnost pri dimenzijskih tabelah je tabela `Dimenzija_obdobje`, ki opredeljuje izbrano časovno dimenzijo poročanja. Atributi tabele opisujejo raven agregiranosti po-



Slika 2.3 Diagram podatkovnega skladišča namišljenega podjetja.

datkov (npr. leto, kvartal, mesec), ključ sestavljen iz kombinacij atributov pa omogoča hierarhijo v dimenziji. Hierarhija je struktura v dimenziji, sestavljena iz zaporedja posameznih ravni, ki so med seboj povezane z relacijo nadrejenosti (npr. leto je nadrejeno kvartalu). Hierarhija omogoča prehajanje med različnimi ravnmi agregiranosti, t.i. vrtanje dol (angl. drill down).

Tabela `Dimenzija_obdobje` je indeksirana s sestavljenim ključem (`Leto`, `Kvartal`, `Mesec`) in vsebuje enoličen atribut `Obdobje_ID`, ki ga predstavlja zaporedna številka zapisa. Preostale dimenzijske tabele so indeksirane po primarnem ključu, ki je tudi zaporedna številka zapisa v tabeli. Tabela `Dejstvo_finance` pa je indeksirana s sestavljenim ključem (`Obdobje_ID`, `Projekt_ID`, `Naročnik_ID`, `Vodja_projekta_ID`, `Tip_storitve_ID`).

2.5 Polnjenje podatkovnega skladišča

Podatkovno skladišče je potrebno napolniti s podatki iz transakcijske podatkovne baze. Pri tem je potrebno ciljna polja v tabelah podatkovnega skladišča povezati z izvornimi podatki. Polnjenje opravlja transformacija, ki v primeru enake zgradbe podatkov v podatkovnem skladišču in transakcijski podatkovni bazi podatke le prekopira, sicer pa jih s

pomočjo vnaprej definiranih pravil ustrezno preoblikuje. Zaradi zagotavljanja integritete podatkov ter minimalnega vpliva polnjenja na delo uporabnikov se to običajno opravlja v času minimalne aktivnosti uporabe informacijskega sistema. Zato sem se v svojem primeru odločil, da se proces polnjenja podatkovnega skladišča izvaja enkrat dnevno v nočnih urah.

Transformacije polj pri polnjenju dimenzijskih tabel prikazuje tabela 2.3. Tabela `Dimenzija_projekt` vsebuje seznam projektov iz tabele `Projekt`, hrani pa atributa `Projekt_ID` in `Ime`. Podobno tudi tabeli `Dimenzija_naročnik` in `Dimenzija_tip_storitve` hranita identifikacijsko številko zapisa in pripadajoče ime iz izvorne baze. Dimenzijska tabela Dimenzija_vodja_projekta se od omenjenih razlikuje po tem, da se polje Priimek_in_ime sestavi iz polj Priimek in Ime iz tabele `Oseba`. Posebnost je ponovno tabela `Dimenzija_obdobje`, ki se napolni s poljubnim številom zapisov, ti pa predstavljajo mesece, za katere se lahko nato napolni tabelo dejstev.

Izvorna tabela	Izvorno polje	Ciljna tabela	Ciljno polje
Projekt	Projekt_ID	Dimenzija_projekt	Projekt_ID
Projekt	Ime	Dimenzija_projekt	Ime
Naročnik	Naročnik_ID	Dimenzija_naročnik	Naročnik_ID
Naročnik	Ime	Dimenzija_naročnik	Ime
Vodja_projekta	Oseba_ID	Dimenzija_vodja_projekta	Vodja_projekta_ID
Oseba	(pravilo)	Dimenzija_vodja_projekta	Priimek_in_ime
Tip_storitve	Tip_storitve_ID	Dimenzija_tip_storitve	Tip_storitve_ID
Tip_storitve	Naziv	Dimenzija_tip_storitve	Naziv
(pravilo)	(pravilo)	Dimenzija_obdobje	Leto
(pravilo)	(pravilo)	Dimenzija_obdobje	Kvartal
(pravilo)	(pravilo)	Dimenzija_obdobje	Mesec

Tabela 2.3 Transformacije polj pri polnjenju dimenzijskih tabel podatkovnega skladišča oglaševalskega podjetja.

Transformacije polj pri polnjenju tabele dejstev `Dejstvo_Finance` prikazuje tabela 2.4. Pred tem je potrebno opraviti polnjenje dimenzijskih tabel, da se lahko naredi sklic nanje. Polja `Projekt_ID`, `Naročnik_ID` in `Projekt_ID` se iz tabele `Projekt` le prekopirajo. Podobno velja za polje `Vodja_projekta_ID`, ki se prekopira iz tabele `Vodja_projekta`. Pri tem se privzema, da je v danem trenutku aktiven le en vodja na projektu, za kar

poskrbi vnosna kontrola pri uporabniškem vmesniku. Preostala polja v tabeli dejstev se polnijo s pomočjo pravil. Polje `Obdobje_ID` predstavlja sklic na dimenzijsko tabelo `Dimenzija_obdobje`, vrednost atributa pa predstavlja mesec, kateremu pripadajo vrednosti polj `Prihodki`, `Stroški`, `Rezultat` in `KPI`. Polje `Prihodki` predstavlja vsoto zneskov vseh postavk na izdanih računih za določen projekt v izbranem mesecu, podobno pa polje `Stroški` predstavlja vsoto vseh postavk na prejetih računih za določen projekt v izbranem mesecu. Polje `Rezultat` se izračuna kot razlika med `Prihodki` in `Stroški`, polje `KPI` pa se izračuna po pravilu, da se v primeru negativne vrednosti polja Rezultat vrne vrednost 0, sicer pa predstavlja razmerje med vrednostima polj `Rezultat` in `Prihodki`.

Izvorna tabela	Izvorno polje	Ciljno polje
(pravilo)	(pravilo)	Obdobje_ID
Projekt	Projekt_ID	Projekt_ID
Projekt	Projekt_ID	Projekt_ID
Projekt	Naročnik_ID	Naročnik_ID
Vodja_projekta	Oseba_ID	Vodja_projekta_ID
Projekt	Tip_storitve_ID	Tip_storitve_ID
Postavka_na_izdanem_računu	(pravilo)	Prihodki
Postavka_na_prejetem_računu	(pravilo)	Stroški
(pravilo)	(pravilo)	Rezultat
(pravilo)	(pravilo)	KPI

Tabela 2.4 Transformacije polj pri polnjenju tabele dejstev `Dejstvo_Finance` oglaševalskega podjetja.

Ker je bil cilj diplomske naloge spremljanje izvajanja poizvedb in ne implementacija transakcijske podatkovne baze ter pripadajočega polnjenja, sem podatkovno skladišče polnil s testnimi podatki. V ta namen sem uporabil proceduri `ETL_Dejstvo_finance` in `ETL_Dimenzije`, ki napolnita tabele z izbranim številom zapisov. Seznam parametrov, ki jih za izvajanje potrebujeta obe proceduri, prikazujeta tabeli 2.5 in 2.6.

Procedura `ETL_Dimenzije` napolni dimenzijske tabele z izbranim številom zapisov, kjer primarni ključ predstavlja zaporedna številka zapisa, ki se obenem vpiše tudi v polje `Ime`, `Priimek_in_ime` oziroma `Naziv`. Primer izpisa podatkov, ki jih generira omenjena procedura, je prikazan v tabeli 2.7. Posebnost je polnjenje tabele `Dimenzija_obdobje`,

Parameter	Opis
@ParamLetoOd	začetno leto
@ParamLetoDo	zaključno leto
@ParamStZapNaLeto	število zapisov na leto

Tabela 2.5 Seznam parametrov in pripadajočih opisov pri proceduri `ETL_Dejstvo_finance`.

Parameter	Opis
@ParamLetoOd	začetno leto
@ParamLetoDo	zaključno leto
@Param_st_p	število projektov
@Param_st_n	število naročnikov
@Param_st_v	število vodij projektov
@Param_st_ts	število tipov storitev

Tabela 2.6 Seznam parametrov in pripadajočih opisov pri proceduri `ETL_Dimenzije`.

v katero se vnese zapise za vse mesece, ki so vsebovani znotraj let podanih v časovnih parametrih. Oblika podatkov, ki jih generira ta procedura je prikazana v tabeli 2.8.

Naročnik_ID	Ime
1	Naročnik 1
2	Naročnik 2
3	Naročnik 3
4	Naročnik 4
5	Naročnik 5
6	Naročnik 6

Tabela 2.7 Primer polnjenja tabele `Dimenzija_naročnik` s testnimi podatki.

Procedura `ETL_Dejstvo_finance` napolni tabelo dejstev z naključno kombinacijo ključev do atributov v dimenzijskih tabelah, pri čemer ignorira morebitne duplikate atributov. Pripadajoče numerične vrednosti meritev v poljih `Prihodki` in `Stroški` napolni z naključnimi vrednostmi, polji `Rezultat` in `KPI` pa izračuna po že opisanih pravilih.

Obdobje_ID	Leto	Kvartal	Mesec
1	2001	1	1
2	2001	1	2
3	2001	1	3
4	2001	2	4
5	2001	2	5
6	2001	2	6

Tabela 2.8 Primer polnjenja tabele `Dimenzija_obdobje` s testnimi podatki.

3 Metode spremljanja izvajanja poizvedb

3.1 Odkrivanje zmogljivostnih težav

Predpogoj za uspešno rabo informatike v podjetju je kakovostno opravljanje storitev informacijskega sistema. To v praksi pomeni hiter odzivni čas pri servisiranju uporabniških zahtev in visoka razpoložljivost ter zanesljivost storitev. Z velikostjo informacijskega sistema naraščajo možnosti za nastanek zmogljivostnih težav, prav tako pa se z velikostjo le-tega veča njegova pomembnost v poslovnih procesih. Zaradi vsega naštetega igra v velikih informacijskih sistemih odprava morebitnih zmogljivostnih težav ključno vlogo.

Do zmogljivostnih težav informacijskega sistema je prišlo tudi na primeru namišljenega oglaševalskega podjetja. Uporabniki informacijskega sistema so namreč skrbnikom podali pritožbe o dolgih odzivnih časih in počasnih izvajanjih opravil, ki jih potrebujejo pri svojem delu. To je skrbnike pripeljalo do suma, da se razlog za težave skriva v preobremenjenosti sistemskih virov podatkovnega strežnika. Za potrditev oziroma zavrnitev svojih sumov potrebujejo orodja, s katerimi bi lahko analizirali izvajanje opravil na strežniku. Ker nadzor izvajanja opravil ne sme po nepotrebnem obremenjevati podatkovnega strežnika, skrbniki uporabijo avtomatski zagon in zaustavitev spremljanja. Takšna

avtomatizacija spremljanje sproži le v primeru, ko je presežen izbrani prag zasedenosti sistemskih virov, in ga zaustavi, ko zasedenost pade pod ta prag.

Skrbniki na ta način pridobijo seznam poizvedb, ki so se izvajale na strežniku ob preobremenitvi in pri tem zasegle največ sistemskih virov. Seznam pošljejo razvijalcem, ki navedene poizvedbe skušajo optimizirati. V kolikor je optimizacija možna in hkrati dovolj učinkovita, da odpravi počasnost, nad katero so se pritoževali uporabniki, razvijalci skrbnikom predajo optimizirane verzije poizvedb. V nasprotnem primeru za odpravo zmogljivostnih težav predlagajo strojno nadgradnjo podatkovnega strežnika.

V nadaljevanju poglavja sem opisal tri orodja, ki nastopajo v procesu odprave zmogljivostnih težav strežnika. Ta orodja so:

- Performance Logs and Alerts, katerega naloga je odkrivanje časovnih intervalov, pri katerih pride do preobremenitve izbranih sistemskih virov,
- `sp_trace`, ki spremlja izvajanje poizvedb, pri tem pa zajema podatke o porabi sistemskih virov,
- načrt izvrševanja poizvedb, ki prikazuje zgradbo poizvedb.

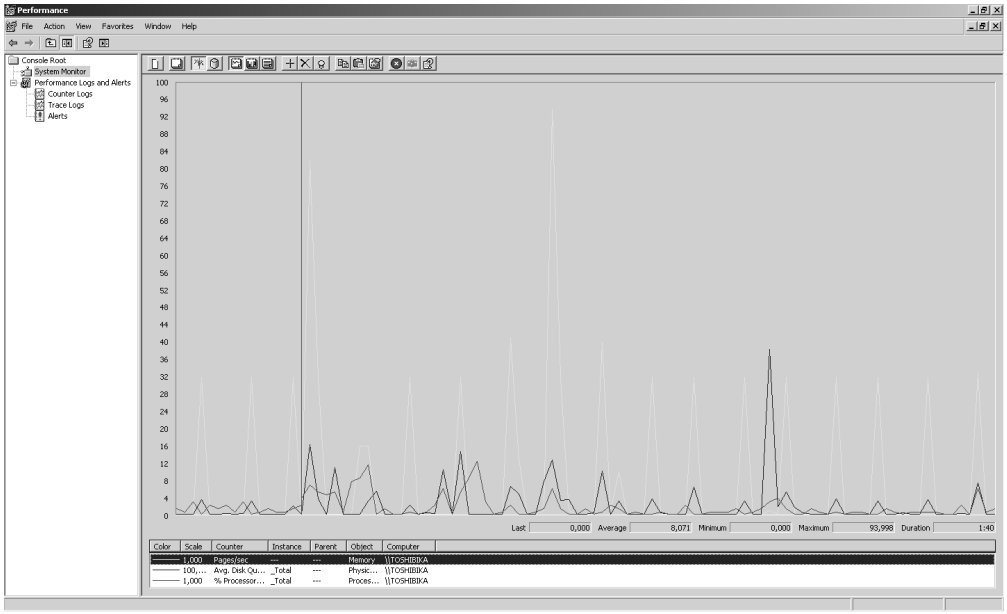
3.2 Zasedenost sistemskih virov

Performance Logs and Alerts je monitor[1] v operacijskem sistemu Windows XP, ki na podlagi vnajprej definiranih pravil spremlja zasedenost sistemskih virov računalnika. Zajete podatke lahko shranjuje v dnevniško datoteko (angl. log file) oziroma jih v realnem času prikazuje v orodju System Monitor, ki je prikazan na sliki 3.1. Orodje omogoča prikaz zajetih podatkov v grafični oziroma tabelarični obliki. Pri grafičnem prikazu se poleg grafa izpisuje še minimalna, maksimalna in povprečna vrednost zasedenosti izbranega sistemskega vira.

S pomočjo podatkov, ki jih System monitor zbira, bi bilo možno avtomatizirati opravila, ki jih potrebujemo pri nadaljnjem spremljanju zmogljivostnih težav, žal pa orodje ne omogoča izvoza in shranjevanja podatkov, zato ga v diplomski nalogi nisem uporabil. Za spremljanje sem posledično uporabil osnovni servis Performance Logs and Alerts, ki ima možnost shranjevanja meritev v:

[1] Monitorji so orodja, realizirana v strojni ali programski opremi, namenjena izvajanju meritev, s katerimi pridobimo podatke o zmogljivostnih lastnostih opazovanega računalniškega sistema.

- tekstovno datoteko,
- binarno datoteko,
- podatkovno bazo.



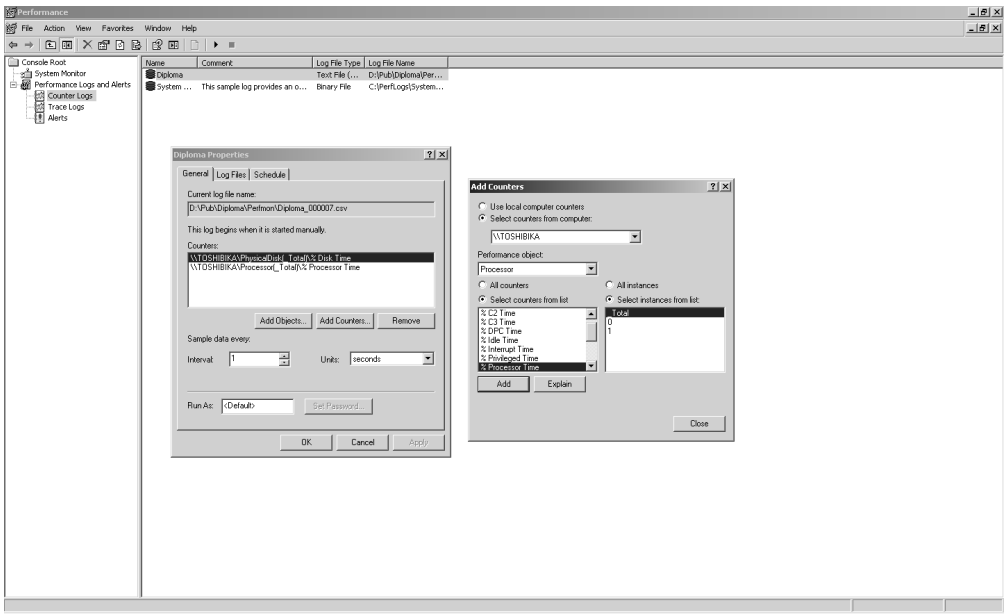
Slika 3.1 Grafični prikaz zajetih podatkov v orodju System Monitor.

V diplomski nalogi sem opazoval zasedenost CPE in trdega diska, pri tem pa sem spremljal sledeče domene[2]:

- delež časa, v katerem CPE izvaja ukaze (domena Processor:% Processor Time) in
- delež časa, v katerem je trdi disk prost (domena PhysicalDisk:% Idle Disk Time).

Nastavitve monitorja sem izbral s pomočjo uporabniške konzole Microsoft Management Console, ki je prikazana na sliki 3.2. Preko konzole sem nastavil spremljanje zgoraj omenjenih domen, podatki pa so se zajemali vsako sekundo.

[2]Z izrazom domena poimenujemo vse podatke, ki jih lahko zajemamo z monitorjem; to so lahko na primer podatki o stanju sistema, času dogodka, številu dostopov do diska itd.



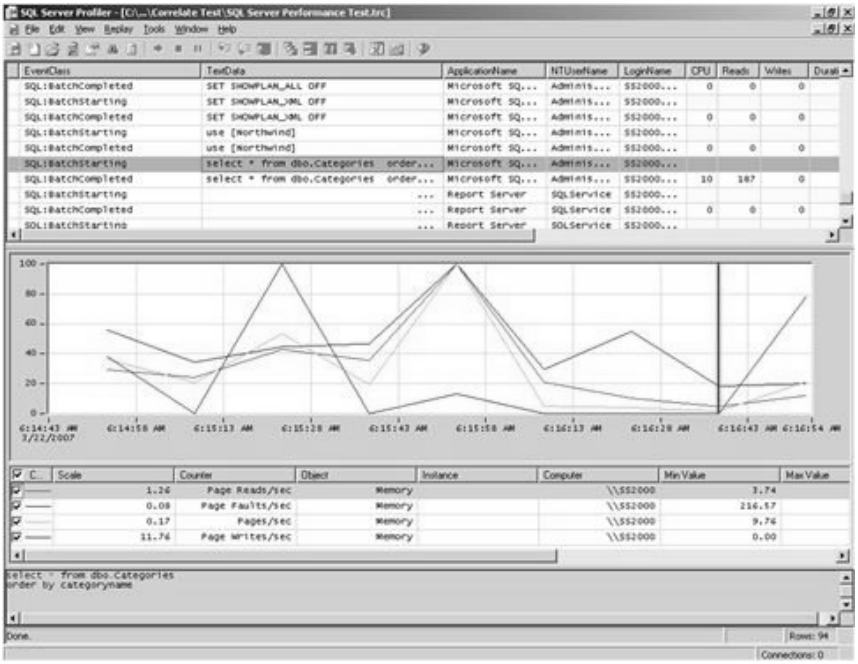
Slika 3.2 Prikaz izbora nastavitev servisa Performance Logs and Alerts.

3.3 Sledenje izvajanja poizvedb

Zbiranju informacij o izvajanju poizvedb na podatkovnem strežniku služijo orodja za sledenje izvajanja poizvedb. Microsoft SQL Server v ta namen uporablja množico baznih procedur (angl. stored procedures) s skupno predpono `sp_trace`. Njihove funkcionalnosti so dostopne tako preko klicev procedur iz orodja SQL Server Management Studio, kot tudi preko orodja SQL Server Profiler, ki je prikazano na sliki 3.3. Ker SQL Server Profiler ni vključen v različico podatkovnega strežnika Express Edition, ki jo uporabljam pri diplomski nalogi, sem sledenje izvajanja poizvedb realiziral s pomočjo klicev procedur iz orodja SQL Server Management Studio.

Moj namen je spremljanje izvajanja tistih poizvedb, ki povzročajo preobremenjenost podatkovnega strežnika, zato zajemam predvsem podatke o porabi sistemskih virov. Na enak način lahko skrbniki identificirajo tiste poizvedbe, ki prekomerno obremenjujejo sistem. Za definicijo in uporabo sledenja so na voljo naslednje bazne procedure:

- `sp_trace_create`: vzpostavi sledenje in določi lokacijo za shranjevanje izpisa,
- `sp_trace_setevent`: ureja kombinacije dogodkov in atributov, ki jih želimo spremljati,



Slika 3.3 SQL Server Profiler.

- `sp_trace_setfilter`: filtrira izpis po izbranih stolpcih,
- `sp_trace_setstatus`: požene, ustavi in zbriše izbrano sled.

Podatke, zbrane z omenjenimi procedurami, sem s funkcijo `fn_trace_gettable` uvozil v tabelo, od koder so mi bile na voljo za nadaljnjo analizo. Izpis iz tovrstne tabele je prikazan na sliki 3.4.

	Opis	Trajanje izvajanja (ms)	Trajanje uporabe CPE (ms)	Št. logičnih branj
1	DECLARE @rc int DECLARE @TraceID int DECLARE @M...	2	0	0
2	CHECKPOINT DBCC DROPCLEANBUFFERS EXEC Po...	19047	2703	17778

Slika 3.4 Primer izpisa generiranega s pomočjo funkcije `fn_trace_gettable`.

Pri sledenju sem opazoval dogodek SQL:BatchCompleted, ki se zgodi, kadar se zaključi izvajanje paketa SQL stavkov (angl. batch) [4]. To v diplomski nalogi predstavlja enkratni zagon poizvedbe. Izvorna koda, ki vzpostavi vse potrebne parametre za zagon sledenja poizvedb, pa je prikazana v izpisu 3.1. Lastnosti dogodka, ki sem jih opazoval, so:

- tekstovni opis dogodka (lastnost TextData),

- trajanje izvajanja poizvedbe v mikrosekundah (lastnost Duration),
- trajanje uporabe CPE v milisekundah (lastnost CPU),
- število logičnih branj[3] (lastnost reads).

Izpis 3.1 Koda za zagon sledenja izvajanja poizvedb.

```
DECLARE @rc int, @TraceID int, @on bit
DECLARE @MaxFileSize bigint, @EndTime datetime
DECLARE @OutputFileName nvarchar(256)

SET @MaxFileSize = 1
SET @OutputFileName = 'C:\\trace'
    + CONVERT(varchar(20), getdate(),112)
    + REPLACE(CONVERT(varchar(20), getdate(),108),':','')
SET @EndTime = DATEADD(s, 30, getdate())

EXEC @rc = sp_trace_create @TraceID output,
    0, @OutputFileName, @MaxFileSize, @EndTime

SET @on = 1

EXEC sp_trace_setevent @TraceID, 12, 1,  @on
EXEC sp_trace_setevent @TraceID, 12, 13, @on
EXEC sp_trace_setevent @TraceID, 12, 16, @on
EXEC sp_trace_setevent @TraceID, 12, 17, @on
EXEC sp_trace_setevent @TraceID, 12, 18, @on

EXEC sp_trace_setstatus @TraceID, 1
```

[3]Pri branju in pisanju se prenašajo 8KB velike strani [5]. Število logičnih branj v primeru zadetka strani v podatkovnem izravnalniku (angl. data buffer) predstavlja število branj strani iz glavnega pomnilnika, v primeru zgrešitve pa število branj iz trdega diska. Število fizičnih pisanj pa predstavlja število pisanj na trdi disk.

3.4 Načrt izvrševanja poizvedb

Načrt izvrševanja poizvedbe (angl. execution plan) prikazuje zaporedje korakov, ki zahtevajo dostop do podatkov, pri tem pa za vsak korak prikaže količino potrebnih operacij za izvršitev [6]. Proces izvrševanja poizvedbe je sestavljen iz zaporedja sledečih korakov:

1. razčlenjevalnik poizvedb (angl. query parser) preveri sintaktično pravilnost poizvedbenega niza (angl. query string), ki je bil podan v izvrševanje podatkovnemu strežniku ter vrne razčlenitveno drevo (angl. parse tree), ki poda potrebne korake za izvršitev poizvedbe;

2. algebrator (angl. algebrizer) iz razčlenitvenega drevesa identificira objekte v podatkovni bazi in tipe funkcij, ki nastopajo v poizvedbi ter vrne procesno drevo poizvedbe (angl. query processor tree);

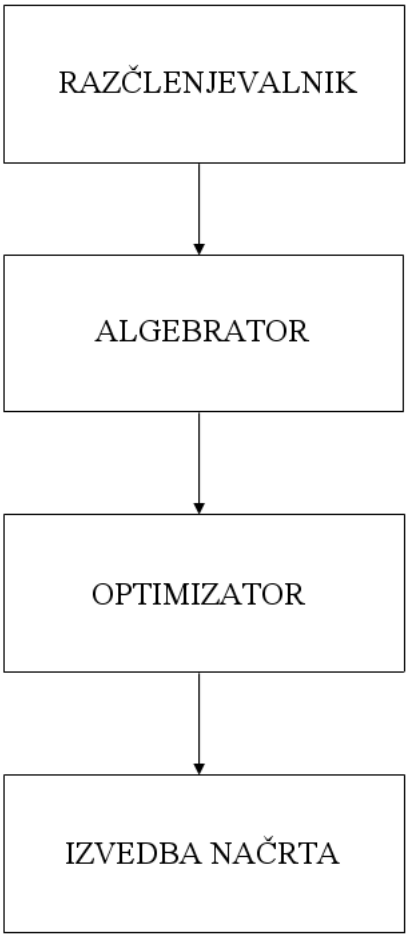
3. procesno drevo obdela optimizator poizvedb (angl. query optimizer), katerega naloga je, da na podlagi obteževanja posameznih elementov drevesa med vsemi možnimi načini izvedbe najde najhitrejšo in vrne načrt izvršenja;

4. izvedba načrta je sprehod po vnaprej izbranih podatkovnih strukturah, urejanje podatkov in njihova dostava odjemalcu.

Opisan proces je prikazan na sliki 3.5. Načrt izvrševanja je torej poskus optimizatorja, da izračuna najbolj učinkovito implementacijo zahteve podane s poizvedbo. Obstajata dve vrsti načrta, ocenjeni (angl. estimated) in dejanski (angl. real). Ocenjeni je rezultat analize podatkovnih struktur, po katerih se bo izvajala poizvedba, ne da bi bila dejansko izvedena. Dejanski je rezultat meritev izvedbe poizvedbe na podlagi ocenjenega načrta.

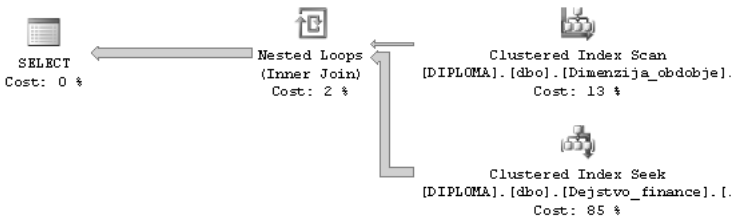
Načrt izvrševanja se lahko predstavi v tabelarični ali grafični obliki. V diplomski nalogi sem uporabil slednjega. Grafična predstavitev izriše drevesno strukturo, v kateri koren predstavlja končni rezultat poizvedbe, preostala vozlišča pa operatorje, ki nastopajo pri njenem izvajanju. Za vsako vozlišče je izpisan atribut Cost, ki prikazuje oceno deleža časa izvajanja operatorja v razmerju z izvajanjem celotne poizvedbe. Primer grafične predstavitve načrta je prikazan na sliki 3.6. Opisi posameznih operatorjev v drevesni strukturi, ki nastopajo v testnih bremenih diplomske naloge, pa so navedeni v tabeli 3.1.

Ob premiku kurzorja nad operator oziroma povezavo v grafičnem prikazu načrta izvrševanja se prikaže pojavno okno (angl. pop-up window) z dodatnimi informacijami.



Slika 3.5 Proces izvrševanja poizvedbe.

V primeru povezave ta prikazuje količino podatkov, ki se prenaša med dvema operatorjema, kar odraža tudi njena debelina. Zgled pojavnega okna je prikazan na sliki 3.7.



Slika 3.6 Načrt izvrševanja poizvedbe v grafični obliki.

Operator	Opis
Compute Scalar	izračun sklarane vrednosti
Clustered Index Scan	metoda iskanja v indeksu
Clustered Index Seek	metoda iskanja v indeksu
Nested Loops	stik med tabelami
Sort	sortiranje zapisov
Stream Aggregate	razvrstitev zapisov v razrede na podlagi enega ali več atributov ter pripadajoč izračun aritmetičnih izrazov

Tabela 3.1 Opis nekaterih operatorjev iz načrta izvrševanja.

Actual Number of Rows	1000000
Estimated Number of Rows	1
Estimated Row Size	28 B
Estimated Data Size	28 B

Slika 3.7 Količina prenešenih podatkov med dvema vozliščema.

4 Izvedba spremljanja izvajanja poizvedb

4.1 Določitev testnih bremen

Problematiko odkrivanja in odpravljanja zmogljivostnih težav sem si na primeru namišljenega oglaševalskega podjetja zamislil na sledeči način. Monitor Performance Logs and Alerts neprestano beleži delež časa, v katerem CPE izvaja ukaze in delež časa, v katerem je trdi disk prost[1] in rezultate meritev shranjuje v posebno dnevniško tabelo v podatkovni bazi. Prožilec[2] (angl. trigger) pri vnosu v to tabelo preverja, če je vrednost meritve za izbrano domeno presegla prag obremenitve, ki ga določimo kot skrbniki. V primeru, da se prag preseže, prožilec samodejno sproži sledenje izvajanja poizvedb s pomočjo baznih procedur `sp_trace` in pošlje skrbnikom elektronsko pošto z obvestilom o preobremenitvi sistemskih virov. Bazne procedure za čas, ki ga določijo skrbniki, spremljajo izvajanje poizvedb. Pri tem podatke zapisuje v podatkovno bazo, od koder so na voljo za nadaljnjo analizo. Avtomatskega proženja in pošiljanja elektronskih obvestil

[1] Obe domeni sta predstavljeni v podpoglavju 3.2.

[2] Prožilec je dejanje, ki se izvede ob določenih pogojih in sproži izvedbo neke akcije ali operacije, v primeru podatkovnega strežnika to pomeni zagon SQL kode.

v diplomski nalogi nisem realiziral, ker sem se osredotočil na rezultate meritev in ne na avtomatizacijo skrbniških opravil.

Po analizi zgoraj omenjenih podatkov pridemo do seznama poizvedb, ki so v času povečane obremenitve podatkovnega strežnika zasegle največ izbranih sistemskih virov. Ta seznam se nato posreduje razvijalcem, ti pa poskušajo optimizirati poizvedbe, ki so zasegle največ sistemskih virov. Pri optimizaciji [7] jim je v pomoč načrt izvrševanja poizvedb, ker jim omogoča vpogled v zgradbo posameznih poizvedb. Popravljene verzije poizvedb se nato predajo nazaj skrbnikom, ki jih namestijo na podatkovni strežnik, tam pa so na voljo uporabnikom.

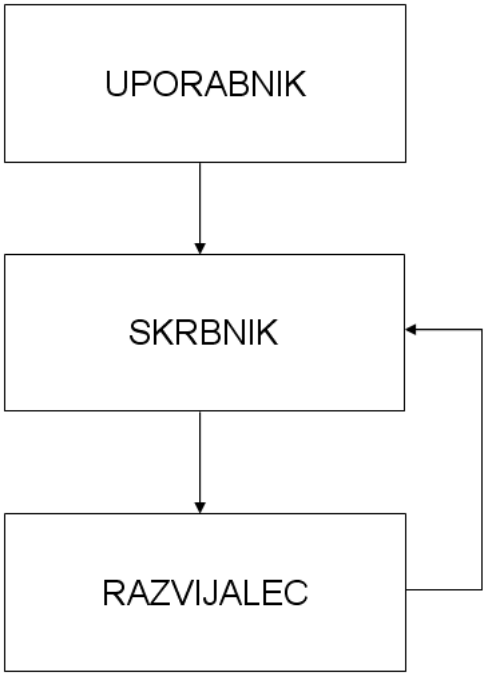
Po namestitvi se z orodjem `sp_trace` izvedejo ponovne meritve in primerjajo zmogljivostne lastnosti obeh verzij. Če se oceni, da je pohitritev popravljenih verzij poizvedb dovolj velika za razrešitev zmogljivostnih težav, se popravljena verzija ohrani. V primeru hitrejšega izvajanja popravljene verzije, ki pa hkrati ni dovolj velika za želeno razrešitev zmogljivostnih težav, je običajna odločitev nakup zmogljivejše strojne opreme. Popravljena verzija se v tem primeru ohrani. V primeru, ko pa je popravljena verzija počasnejša od prvotne, se le-ta zamenja s prvotno in odločitev za nadgradnjo strojne opreme je v tem primeru neizogibna.

Na sliki 4.1 so prikazani udeleženci opisanega procesa odprave zmogljivostnih težav. V svoji diplomski nalogi sem privzel, da so skrbniki na seznam kandidatov za optimizacijo uvrstili tri poizvedbe:

- kazalnik KPI,
- prihodki na projektih za tekoče in prejšnje leto,
- rezultat po naročnikih med leti 2006 in 2010.

Omenjene poizvedbe sem poganjal na podatkovnem strežniku[3], na katerem sem izvajal meritve. V nadaljevanju predstavljam okoliščine, v katerih sem izvajal meritve, rezultate meritev ter njihovo interpretacijo. Pri interpretaciji sem skušal podati odgovor na vprašanje, ali je za odpravo zmogljivostnih težav podatkovnega strežnika potrebno nadgraditi strojno opremo ali ne.

[3] Podatkovni strežnik sem natančneje predstavil v podpoglavju 2.2.



Slika 4.1 Udeleženci procesa odprave zmogljivostnih težav.

4.2 Vzpostavitev začetnega stanja

Za verodostojno ponazoritev izsledkov meritev sem moral vzpostaviti začetno stanje, pri čemer pretekle meritve niso smele vplivati na rezultat meritve, ki se je trenutno izvajala. Za ponastavitev okolja pred vsako meritvijo sem pred vsakim zagonom poizvedbe najprej izpraznil vsebino podatkovnega izravnalnika (angl. data buffer) na podatkovnem strežniku. Na čas izvajanja vpliva tudi izdelava načrta poizvedbe, ki se izdela ob njenem prvem zagonu. Pri nadaljnjih zagonih iste poizvedbe podatkovni strežnik najprej pogleda v predpomnilnik načrtov (angl. plan cache), če zanjo obstaja veljaven načrt [8]. V primeru obstoja se načrt uporabi, sicer pa se ponovno izdela.

Za praznjenje izravnalnika sem uporabil kombinacijo ukazov `CHECKPOINT` in `DBCC DROPCLEANBUFFERS`. `CHECKPOINT` umakne vse umazane bloke iz izravnalnika. Umazani bloki so tisti, kjer vsebina bloka v izravnalniku ni skladna z vsebino pripadajočega bloka na trdem tisku. V nasprotnem primeru gre za čiste bloke, te pa iz izravnalnika umakne ukaz `DBCC DROPCLEANBUFFERS`. S klicem obeh ukazov pred začetkom meritve sem dosegel izpraznitev celotnega izravnalnika.

Pri ponovitvah meritev posameznih poizvedb sem privzel, da ostajajo pripadajoči načrti izvrševanja veljavni. Pri tem pa se prva meritev razlikuje od ostalih, ker je pri prvem zagonu poizvedbe vključena izdelava načrta. Problem enakovrednosti meritev bi lahko rešil z zagonom ukaza `DBCC FREEPROCCACHE` pred vsakim zagonom poizvedbe. Ta bi zbrisal vse obstoječe načrte poizvedb iz predpomnilnika načrtov. Ker pa sem želel pri meritvah ignorirati izdelavo načrta, sem začel z njimi šele pri drugem zagonu poizvedbe, ko je bil načrt že izdelan.

4.3 Meritve na testnih bremenih

V časovnem intervalu, ko je podatkovni strežnik preobremenjen, zasegajo v podpoglavju 4.1 naštete poizvedbe opazno količino sistemskih virov. Razlog za požrešnost poizvedb je relativno velika količina procesiranja, ki ga algoritmi opravijo. Uporabljeni algoritmi opravljajo bolj ali manj zahtevne izračune nad veliko množico podatkov iz podatkovnega skladišča, pri čemer lahko uspešna optimizacija algoritma predstavlja opazen prihranek pri izrabi sistemskih virov.

V podpoglavjih sem tako sistematično obdelal vse tri poizvedbe, vsako pa sem predstavil v dveh verzijah:

- verzija A predstavlja različico, pri kateri opazimo veliko izrabo sistemskih virov,
- verzija B pa predstavlja poskus optimizacije poizvedbe s strani razvijalcev.

V nadaljevanju predstavljam izbrane poizvedbe, rezultate meritev na vseh treh in interpretacijo dobljenih izsledkov. Meritev in razlage sem se lotil sistematično na način opisan v nadaljevanju. Vsako poizvedbo sem najprej opisal z uporabniškega vidika in omenil morebitne posebnosti. Nato sem opredelil predmet opazovanja v poizvedbi in predstavil izvorno kodo verzije A. Pri tem sem prikazal primer opravljene meritve s Performance Logs and Alerts. Sledi tabela z meritvami opravljenimi z `sp_trace` za verzijo A, kjer sem opravil pet meritev in izračunal povprečno vrednost le-teh. Meril sem čas celotnega izvajanja poizvedbe, trajanje uporabe CPE pri tem in število logičnih branj.

Primerjal sem tudi posamezne meritve med seboj ter izračunal odstopanja od povprečnih vrednosti. Nato sem na podoben način predstavil verzijo B. Sledi primerjava rezultatov meritev obeh verzij opravljenih z `sp_trace`, kjer v tabeli stolpec B/A pri-

kazuje razmerje povprečni vrednosti med verzijama. Na podlagi rezultatov meritev na koncu podajam predloge za razrešitev zmogljivostnih težav podatkovnega strežnika.

Pri vsaki poizvedbi sem za verziji A in B opravil pet meritev z orodjem `sp_trace` in shranil dejanski načrt izvrševanja. `Sp_trace` sem prožil ročno in ne s pomočjo prožilcev, kot je to predstavljeno v podpoglavju 4.1. Za ročno proženje sem se odločil zato, ker so me zanimali predvsem rezultati meritev predstavljenih orodij in ne delovanje avtomatskega proženja `sp_trace`. Kljub temu predstavljam zgled izpisa iz dnevniške tabele monitorja Performance Logs and Alerts.

4.3.1 Kazalnik KPI

Kazalnik KPI je poizvedba, ki razkriva ključni performančni indikator poslovanja. Definirali so ga vodilni analitiki v podjetju in prikazuje vrednosti, ki odražajo uspešnost poslovanja različnih tipov storitev za vsa leta, za katera so na voljo podatki. KPI se izračuna po formuli, ki je prikazana v izpisu 4.1. Ta v primeru pozitivnega rezultata poslovanja prikazuje razmerje med rezultati in prihodki, sicer pa vrača vrednost 0. Primer izpisa poizvedbe je prikazan v tabeli 4.1.

Izpis 4.1 Koda za zagon sledenja izvajanja poizvedb.

```
SELECT
    CASE
        WHEN T1.Rezultat>0 THEN (T1.Rezultat/T1.Prihodki)
        ELSE 0
    END
FROM Dejstvo_finance AS T1
```

Tip_storitve_ID	Leto	Povprečni_KPI
1	2010	0,367208
2	2010	0,184273
3	2010	0,732183
4	2010	0,448273
5	2010	0,837421

Tabela 4.1 Prikaz izpisa poizvedbe kazalnik KPI.

Pri testnem bremenu sem opazoval, kako vnaprej izračunano polje vpliva na hitrost izvajanja poizvedbe. Izvorna koda poizvedbe A je prikazana v izpisu 4.2. Meritve monitorja Performance Logs and Alerts, ki sem jih opravil na poizvedbi verzije A, prikazuje tabela 4.2. Rezultati meritev orodja `sp_trace` so prikazani v tabeli 4.3. Po primerjavi rezultatov posameznih meritev iz te tabele sem ugotovil, da razlika med minimalnim in maksimalnim časom izvajanja poizvedbe znaša približno 1,7% povprečnega časa izvajanja petih neodvisnih meritev. Razlika med minimalnim in maksimalnim trajanjem uporabe CPE pa znaša približno 12%.

Izpis 4.2 Izvorna koda poizvedbe `Kazalnik_KPI_A`.

```
SELECT
    T1.Tip_storitve_ID,
    T2.Leto,
    AVG
    (
      CASE
          WHEN T1.Rezultat>0 THEN (T1.Rezultat/T1.Prihodki)
          ELSE 0
      END
    ) AS Povprečni_KPI
FROM Dejstvo_finance AS T1
    INNER JOIN Dimenzija_obdobje AS T2 ON
        T1.Obdobje_ID = T2.Obdobje_ID
GROUP BY T1.Tip_storitve_ID, T2.Leto
```

Denimo, da bi po analizi algoritma ter dnevniške datoteke domnevali, da bi k pohitritvi poizvedbe pripomogla vpeljava novega polja v tabeli `Dejstvo_finance`, ki bi hranila vnaprej izračunano vrednost kazalnika. S tem odpade potreba po sprotnem računanju le tega pri vsakem zagonu poizvedbe, kar sprosti uporabo CPE. Izvorna koda predlagane izboljšave poizvedbe je prikazana v izpisu 4.3, rezultate meritev pa prikazuje tabela 4.4. Po primerjavi rezultatov posameznih meritev iz te tabele sem ugotovil, da razlika med minimalnim in maksimalnim časom izvajanja posodobljene poizvedbe znaša približno 3,2% povprečnega časa izvajanja petih neodvisnih meritev. Razlika med minimalnim in maksimalnim trajanjem uporabe CPE pa znaša približno 13%.

Čas	PhysicalDisk(_Total)% Idle Time	Processor(_Total)% Processor Time
19:46:29,734	99,91	0,00
19:46:31,734	19,55	6,25
19:46:33,734	18,23	12,50
19:46:35,734	18,07	11,71
19:46:37,734	19,23	14,84
19:46:39,734	0,28	4,68
19:46:41,734	0,25	8,59
19:46:43,734	0,27	6,25
19:46:45,734	0,22	1,56
19:46:47,734	0,24	7,03
19:46:49,734	4,79	10,93
19:46:51,734	95,21	0,78

Tabela 4.2 Primer izpisa datoteke z dnevnikom za poizvedbo Kazalnik_KPI_A.

Parameter	Test 1	Test 2	Test 3	Test 4	Test 5	Povprečje
Čas izvajanja (ms)	19124	19176	18986	19316	19013	19123,00
Trajanje uporabe CPE (ms)	3078	2938	3156	2796	2812	2956,00
Št. logičnih branj	17775	17775	17777	17776	17777	17776,00

Tabela 4.3 Rezultat meritev za poizvedbo `Kazalnik_KPI_A`.

Izpis 4.3 Izvorna koda poizvedbe `Kazalnik_KPI_B`.

```
SELECT
    T1.Tip_storitve_ID,
    T2.Leto,
    AVG(T1.KPI) AS Povprečni_KPI
FROM Dejstvo_finance AS T1
    INNER JOIN Dimenzija_obdobje AS T2 ON
        T1.Obdobje_ID = T2.Obdobje_ID
GROUP BY T1.Tip_storitve_ID, T2.Leto
```

Parameter	Test 1	Test 2	Test 3	Test 4	Test 5	Povprečje
Čas izvajanja (ms)	17092	17605	17045	17279	17439	17292,00
Trajanje uporabe CPE (ms)	2562	2375	2719	2594	2625	2575,00
Št. logičnih branj	17392	17392	17392	17136	17390	17340,40

Tabela 4.4 Rezultat meritev za poizvedbo `Kazalnik_KPI_B`.

Medsebojna primerjava osnovne in prenovljene poizvedbe je navedena v tabeli 4.5. Verzija B se je izvajala za skoraj 10% hitreje kot verzija A, pri tem pa je CPE uporabljala za približno 13% manj. Število logičnih branj je pri obeh verzijah ostalo približno enako.

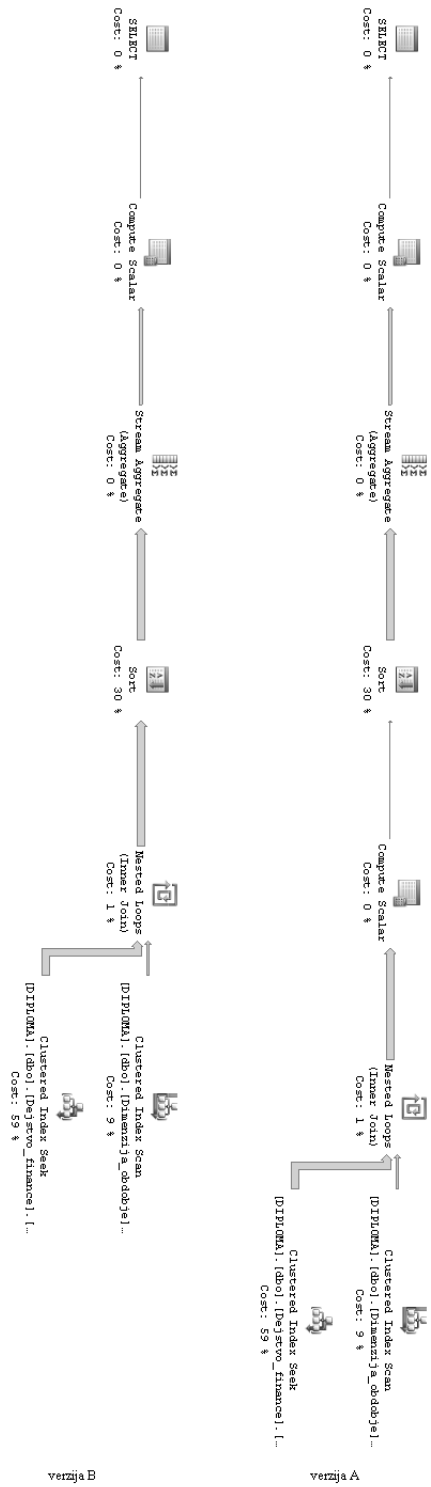
Parameter	B / A
Čas izvajanja (ms)	90,46
Trajanje uporabe CPE (ms)	87,11
Št. logičnih branj	97,55

Tabela 4.5 Primerjava rezultatov meritev za poizvedbi kazalnik KPI.

Na sliki 4.2 sta prikazana oba načrta poizvedb. Načrt verzije B vsebuje eno vozlišče manj kot tisti verzije A, t.j. vozlišče tipa Compute Scalar s ceno 0%. Sicer so cene preostalih vozlišč pri obeh verzijah enake. Najvišjo ceno ima vozlišče tipa Clustered Index Seek, ki dostopa do tabele Dejstvo_Finance, in sicer 59%. Sledi mu vozlišče Sort s ceno 30%, ki med izvajanjem poizvedbe ureja zapise iz tabele Dejstvo_Finance. Vozlišče tipa Clustered Index Scan, ki dostopa do tabele Dimenzija_obdobje, ima ceno 9%, vozlišče tipa Nested Loops pa 1%. Vsota vseh preostalih vozlišč ima ceno približno 1%.

Iz tabele 4.5 je razvidno, da je pohitritev prisotna pri času izvajanja celotne poizvedbe ter trajanju uporabe CPE. To me je pripeljalo do zaključka, da vnaprej izračunano polje pohitri izvajanje poizvedb, ker odpade potreba po sprotnih izračunih. Pohitritev zmanjša število pripadajočih aritmetičnih operacij, ki so potrebne za dokončanje poizvedbe. Preseneča me pa, da se oken v enem vozlišču ne odraža v dejanskem načrtu izvrševanja.

Izboljšava, ki so jo predlagali razvijalci, pohitri poizvedbo na testnem okolju za približno 10%. Skrbnikom bi predlagal, da omenjena izboljšava zadostuje za odpravo zmogljivostnih težav ter da nadgradnja strojne opreme podatkovnega strežnika v tem primeru ni potrebna.



Slika 4.2 Načrt poizvedbe za Kazalnik KPI.

4.3.2 Prihodki na projektih za tekoče in prejšnje leto

Prihodki so vsote postavk na izdanih računih. Podjetje namreč za potrebe poslovnega procesa vodi evidenco izvedenih storitev v obliki projektov, pri čemer se prihodki in stroški na projektih knjižijo preko izdanih in prejetih računov. Poizvedba vrne prihodke na projektih za tekoče in prejšnje leto, pri tem pa se vrednost za tekoče leto določi s pomočjo kode, ki je navedena v izpisu 4.4. Primer izpisa je prikazan v tabeli 4.6.

Izpis 4.4 Nastavitev vrednosti parametra za tekoče leto.

```
DECLARE @Leto int
SET @Leto = DATEPART(year, getdate())
```

Projekt_ID_ID	Leto	Prihodki
1	2009	306.730,93
2	2009	87.218,13
3	2009	236.670,62
1	2010	139.851,61
2	2010	946.881,19
3	2010	213.521,32

Tabela 4.6 Prikaz izpisa poizvedbe prihodki na projektih za tekoče in prejšnje leto.

Pri testnem bremenu sem opazoval, kako uporaba različnih kriterijev pri `WHERE` pogoju SQL stavka `SELECT` vpliva na izbor metode iskanja po indeksu. Izvorna koda poizvedbe A je prikazana v izpisu 4.5, meritve monitorja Performance Logs and Alerts, ki sem jih opravil na poizvedbi verzije A, prikazuje tabela 4.7. Rezultati meritev orodja `sp_trace` izvorne verzije poizvedbe so prikazani v tabeli 4.8. Po primerjavi rezultatov posameznih meritev iz te tabele sem ugotovil, da razlika med minimalnim in maksimalnim časom izvajanja poizvedbe znaša približno 8% povprečnega časa izvajanja petih neodvisnih meritev. Razlika med minimalnim in maksimalnim trajanjem uporabe CPE pa znaša približno 16%.

Denimo da pri analizi algoritma opazijo uporabo logične operacije `OR` pri `WHERE` kriteriju in kot izboljšavo predlagajo uporabo operatorja `BETWEEN`. Izvorna koda predlagane izboljšave poizvedbe je prikazana v izpisu 4.6, rezultate meritev pa prikazuje tabela 4.9. Po primerjavi rezultatov posameznih meritev iz te tabele sem ugotovil, da razlika med

Izpis 4.5 Izvorna koda poizvedbe `Prihodki_na_projektih_za_tekoče_in_prejšnje_leto_A`.

```
SELECT
    T1.Projekt_ID,
    T2.Leto,
    SUM(T1.Rezultat) AS Rezultat
FROM Dejstvo_finance AS T1
    INNER JOIN Dimenzija_obdobje AS T2 ON
        T1.Obdobje_ID = T2.Obdobje_ID
WHERE T2.Leto = @Leto OR T2.Leto = @Leto - 1
GROUP BY T1.Projekt_ID, T2.Leto
```

Čas	PhysicalDisk(_Total)% Idle Time	Processor(_Total)% Processor Time
19:47:12,156	87,72	1,56
19:47:13,156	7,91	8,59
19:47:14,156	2,80	4,68
19:47:15,156	15,05	32,03
19:47:16,156	99,30	0,78

Tabela 4.7 Primer izpisa datoteke z dnevnikom za poizvedbo `Prihodki_na_projektih_za_tekoče_in_prejšnje_leto_A`.

minimalnim in maksimalnim časom izvajanja poizvedbe znaša približno 9% povprečnega časa izvajanja petih neodvisnih meritev. Razlika med minimalnim in maksimalnim trajanjem uporabe CPE pa znaša približno 21%.

Primerjava med izvorno in popravljeno poizvedbo pa je navedena v tabeli 4.10. Verzija B se je v povprečju izvajala za približno 2% hitreje kot verzija A, pri tem pa je CPE uporabljala za približno 3% manj časa. Število logičnih branj je v obeh primerih ostalo približno enako.

Na sliki 4.3 sta prikazana oba načrta poizvedb. Načrta izvrševanja verzij poizvedb imata enako število vozlišč, razlikujeta pa se v tipu vozlišča, ki dostopa do tabele Dimenzija_obdobje. Verzija A uporablja Clustered Index Scan, ki ima ceno 16%, verzija B pa Clustered Index Seek, ki ima ceno 17%. Vozlišče Clustered Index Seek, ki dostopa do tabele Dejstvo_finance, ima pri verziji A ceno 31%, pri verziji B pa 25%. Slednja metoda

Parameter	Test 1	Test 2	Test 3	Test 4	Test 5	Povprečje
Čas izvajanja (ms)	3067	2920	2965	2981	3205	3027,60
Trajanje uporabe CPE (ms)	422	407	359	375	360	384,60
Št. logičnih branj	3353	3606	3624	3632	3632	3569,40

Tabela 4.8 Rezultati meritev za poizvedbo `Prihodki_na_projektih_za_tekoče_in_prejšnje_leto_A`.

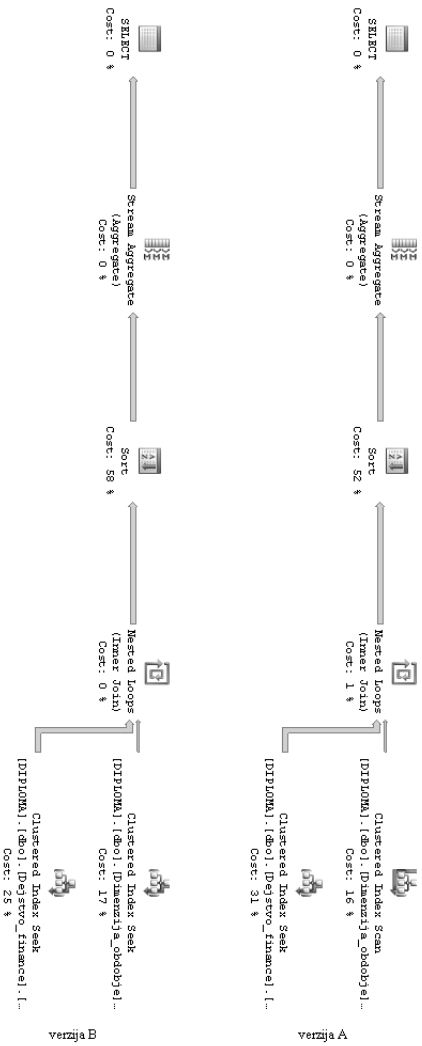
Izpis 4.6 Izvorna koda poizvedbe `Prihodki_na_projektih_za_tekoče_in_prejšnje_leto_B`.

```
SELECT
    T1.Projekt_ID,
    T2.Leto,
    SUM(T1.Rezultat) AS Rezultat
FROM Dejstvo_finance AS T1
    INNER JOIN Dimenzija_obdobje AS T2 ON
        T1.Obdobje_ID = T2.Obdobje_ID
WHERE T2.Leto BETWEEN @Leto - 1 AND @Leto
GROUP BY T1.Projekt_ID, T2.Leto
```

iskanja po indeksu je opazno hitrejša pri tabelah, ki vsebujejo veliko število zapisov [9]. Vozlišče Sort ima pri verziji A ceno 52%, pri verziji B pa ceno 58%. Vsote cen preostalih vozlišč obeh verzij znašajo manj kot 1%.

Iz tabele 4.10 je razvidno, da je bila pohitritev prisotna pri času izvajanja celotne poizvedbe ter trajanju uporabe CPE pri tem. To me je pripeljalo do zaključka, da tovrstna zamenjava operatorja pri `WHERE` pogoju pohitri izvajanje poizvedb. Pohitritev se zgodi zaradi uporabe hitrejše metode iskanja po indeksu. Ker cilj pričujoče diplomske naloge ni optimizacija poizvedb, razlogov, zakaj uporaba operatorja `OR` pri `WHERE` pogoju povzroči uporabo počasnejše metode, nisem raziskoval. Razlika med hitrostjo obeh metod iskanja po indeksu je odvisna tudi od števila zapisov v tabeli, to pa je v tabeli `Dimenzija_obdobje` pogojeno z izbiro zrnatosti podatkovnega skladišča. V primeru povečanja zrnatosti podatkovnega skladišča bi bila pohitritev večja.

Izboljšava, ki so jo predlagali razvijalci, pohitri poizvedbo na testnem okolju za približno 2%. Načrt izvrševanja poizvedb se je izkazal za koristnega, ker sem z njegovo pomočjo odkril uporabo počasnejše metode iskanja po indeksu kot bi jo lahko sicer upo-



Slika 4.3 Načrt poizvedbe za prihodki na projektih za tekoče in prejšnje leto.

Parameter	Test 1	Test 2	Test 3	Test 4	Test 5	Povprečje
Čas izvajanja (ms)	3041	3000	2774	3008	2992	2963,00
Trajanje uporabe CPE (ms)	422	375	344	359	359	371,80
Št. logičnih branj	3377	3630	3631	3631	3631	3580,00

Tabela 4.9 Rezultat meritev za poizvedbo `Prihodki_na_projektih_za_tekoče_in_prejšnje_leto_B`.

Parameter	B A
Čas izvajanja (ms)	97,87
Trajanje uporabe CPE (ms)	96,67
Št. logičnih branj	100,30

Tabela 4.10 Primerjava rezultatov meritev za poizvedbi prihodki na projektih za tekoče in prejšnje leto.

rabil. Skrbnikom bi v tem primeru priporočil namestitev popravljene verzije poizvedbe na podatkovni strežnik, hkrati pa tudi nadgradnjo strojne opreme, ker pohitritev ni dovolj velika, da bi odpravila zmogljivostne težave.

4.3.3 Rezultat po naročnikih med leti 2006 in 2010

Rezultat je razlika med prihodki in stroški, pri čemer prvi predstavljajo vsoto postavk na izdanih računih, drugi pa vsoto postavk na prejetih računih. Poizvedba vrne rezultat poslovanja po naročnikih med leti 2006 in 2010, pri čemer stolpci predstavljajo leta. Primer izpisa je prikazan v tabeli 4.11.

Naročnik_ID	2006	2007	2008	2009	2010
1	51.532,21	234.632,43	122.234,62	321.124,73	272.623,21
2	492.211,23	391.125,33	532.321,84	284.124,19	143.124,94
3	401.231,84	424.123,87	127.841,94	124.834,32	362.381,43
4	25.723,53	46.342,41	33.212,38	42.234,63	37.235,11
5	441.124,42	520.432,02	478.234,73	401,284,32	463.221,01

Tabela 4.11 Prikaz izpisa poizvedbe rezultat pri naročnikih med leti 2006 in 2010.

Pri testnem bremenu sem opazoval zmogljivostne lastnosti operatorja `PIVOT`. Izvorna koda poizvedbe je prikazana v izpisu 4.7. Meritve monitorja Performance Logs and

Alerts, ki sem jih opravil na poizvedbi verzije A, prikazuje tabela 4.12. Rezultati meritev orodja `sp_trace` izvorne verzije poizvedbe so prikazani v tabeli 4.13. Po primerjavi rezultatov posameznih meritev iz tabele sem ugotovil, da razlika med minimalnim in maksimalnim časom izvajanja poizvedbe znaša približno 5% povprečnega časa izvajanja petih neodvisnih meritev. Razlika med minimalnim in maksimalnim trajanjem uporabe CPE znaša približno 20%.

Izpis 4.7 Izvorna koda poizvedbe `Rezultat_pri_naročnikih_med_leti_2006_in_2010_A`.

```
SELECT
    T1.Naročnik_ID,
    SUM(CASE WHEN T2.Leto=2006 THEN T1.Stroški ELSE 0 END)
    AS [2006],
    SUM(CASE WHEN T2.Leto=2007 THEN T1.Stroški ELSE 0 END)
    AS [2007],
    SUM(CASE WHEN T2.Leto=2008 THEN T1.Stroški ELSE 0 END)
    AS [2008],
    SUM(CASE WHEN T2.Leto=2009 THEN T1.Stroški ELSE 0 END)
    AS [2009],
    SUM(CASE WHEN T2.Leto=2010 THEN T1.Stroški ELSE 0 END)
    AS [2010]
FROM Dejstvo_finance AS T1
    INNER JOIN Dimenzija_obdobje AS T2 ON
        T1.Obdobje_ID = T2.Obdobje_ID
WHERE T2.Leto IN (2006, 2007, 2008, 2009, 2010)
GROUP BY T1.Naročnik_ID
```

Denimo da po analizi poizvedbe poskušajo optimizirati kodo z vpeljavo operatorja `PIVOT`, da bi le-ta transponiral leta v stolpce. Izvorna koda poizvedbe je prikazana v izpisu 4.8, rezultate meritev popravljene verzije poizvedbe pa prikazuje tabela 4.14. Po primerjavi rezultatov posameznih meritev iz tabele sem ugotovil, da razlika med minimalnim in maksimalnim časom izvajanja poizvedbe znaša približno 9% povprečnega časa izvajanja petih neodvisnih meritev. Razlika med minimalnim in maksimalnim trajanjem uporabe CPE znaša približno 12%.

Primerjava obeh različic poizvedb je navedena v tabeli 4.15. Verzija B se je izvajala

Čas	PhysicalDisk(_Total)% Idle Time	Processor(_Total)% Processor Time
19:47:53.890	99,98	1,56
19:47:54.890	155.54	1.56
19:47:55.890	66,37	2,34
19:47:56.890	10,81	7,81
19:47:57.890	7,54	7,03
19:47:58.890	8,56	4,68
19:47:59.890	3,94	1,56
19:48:00.890	0,30	3,90
19:48:01.890	0,19	3,90
19:48:02.890	0,17	7,03
19:48:03.890	0,22	3,90
19:48:04.890	5,23	19,53
19:48:04.890	99,92	3,12

Tabela 4.12 Primer izpisa datoteke z dnevnikom za poizvedbo `Rezultat_pri_naročnikih_med_leti_2006_in_2010_A`.

za skoraj 17% počasneje kot verzija A, pri čemer se je trajanje uporabe CPE pri verziji B malenkost zmanjšalo. Število logičnih branj je tudi v tem primeru ostalo približno enako.

Na sliki 4.4 sta prikazana oba načrta poizvedb. Načrt verzije B vsebuje eno vozlišče manj kot tisti verzije A, t.j. vozlišče tipa Compute Scalar s ceno 0%. Sicer so cene preostalih vozlišč pri obeh verzijah enake. Najvišjo ceno ima vozlišče tipa Clustered Index Seek, ki dostopa do tabele Dejstvo_Finance, in sicer 46%. Sledi mu vozlišče Sort s ceno 41%, ki med izvajanjem poizvedbe ureja zapise iz tabele Dejstvo_finance. Vozlišče tipa Clustered Index Seek, ki dostopa do tabele Dimenzija_obdobje, ima ceno 12%, vozlišče tipa Nested Loops pa 1%. Vsota vseh preostalih vozlišč ima ceno manj kot 1%.

Iz tabele 4.15 je razvidno, da je upočasnitev prisotna predvsem pri času izvajanja celotne poizvedbe, čas trajanja uporabe CPE pa ostaja približno enak. To me je pripeljalo do zaključka, da vpeljava operatorja `PIVOT` ne pospeši izvajanja poizvedb, ampak ga celo upočasni za približno 17%. Preseneča me, da ostajajo cene vozlišč v načrtu izvrševanja poizvedb pri obeh verzijah enake, čeprav je opazna razlika v času izvajanja

Parameter	Test 1	Test 2	Test 3	Test 4	Test 5	Povprečje
Čas izvajanja (ms)	9603	9599	9187	9114	9416	9384,20
Trajanje uporabe CPE (ms)	1500	1234	1391	1250	1391	1353,20
Št. logičnih branj	8798	9045	9049	8813	9071	8955,20

Tabela 4.13 Rezultat meritev za poizvedbo `Rezultat_pri_naročnikih_med_leti_2006_in_2010_A`.

Parameter	Test 1	Test 2	Test 3	Test 4	Test 5	Povprečje
Čas izvajanja (ms)	10731	11541	10812	10597	10935	10923,20
Trajanje uporabe CPE (ms)	1360	1453	1297	1312	1234	1331,20
Št. logičnih branj	8961	9219	9217	8983	9237	9123,40

Tabela 4.14 Rezultat meritev za poizvedbo `Rezultat_pri_naročnikih_med_leti_2006_in_2010_B`.

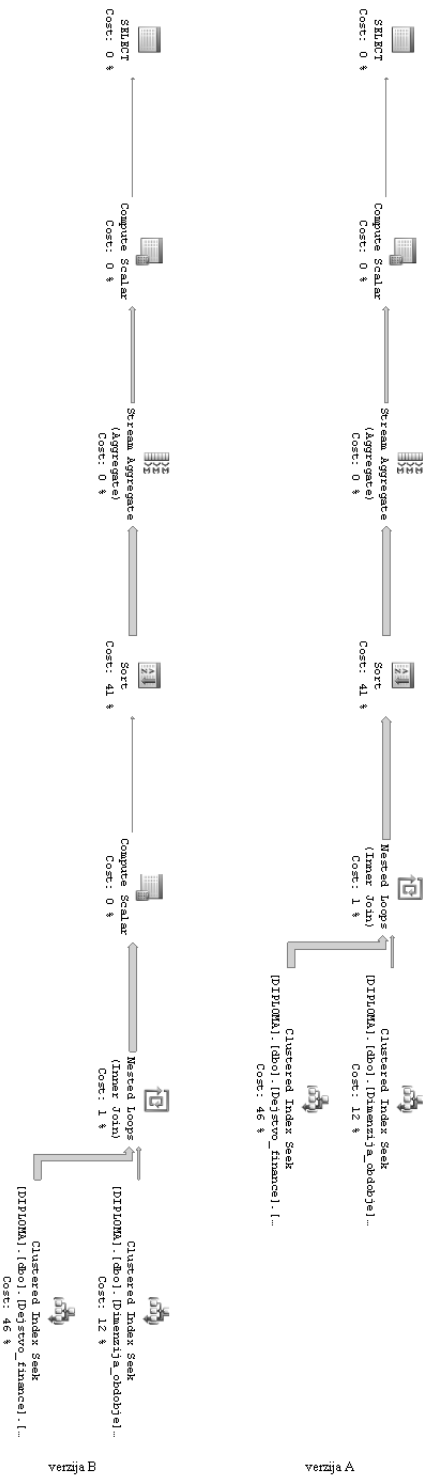
Parameter	B A
Čas izvajanja (ms)	116,64
Trajanje uporabe CPE (ms)	98,37
Št. logičnih branj	101,19

Tabela 4.15 Primerjava rezultatov meritev za poizvedbi rezultat pri naročnikih mede leti 2006 in 2010.

poizvedb. Zaradi povedanega bi z namenom odprave zmogljvostnih težav skrbnikom priporočil nadgradnjo strojne opreme strežnika in odločitev, da obdržijo izvorno različico poizvedbe.

Izpis 4.8 Izvorna koda poizvedbe Rezultat_pri_naročnikih_med_leti_2006_in_2010_B.

```
SELECT Naročnik_ID, [2006], [2007], [2008], [2009], [2010]
FROM
    (
        SELECT T1.Naročnik_ID, T2.Leto, T1.Rezultat
        FROM Dejstvo_finance AS T1
            INNER JOIN Dimenzija_obdobje AS T2 ON
                T1.Obdobje_ID = T2.Obdobje_ID
        WHERE T2.Leto IN (2006, 2007, 2008, 2009, 2010)
    ) AS T1
PIVOT
    (
        SUM(T1.Stroški)
        FOR T1.Leto
        IN ([2006], [2007], [2008], [2009], [2010])
    ) AS pvt
ORDER BY Naročnik_ID
```



Slika 4.4 Načrt poizvedbe za rezultat pri naročnikih med leti 2006 in 2010.

5 Zaključek

5.1 Sklepne ugotovitve

V pričujoči diplomski nalogi sem predstavil uporabo treh orodij (Performance Logs and Alerts, `sp_trace` in načrt izvrševanja poizvedb) pri iskanju vzrokov zmogljivostnih težav na podatkovnem strežniku namišljenega oglaševalskega podjetja. Orodja so mi pomagala poiskati odgovore na vprašanja, ki sem si jih zastavil uvodoma:

- Kako odkriti časovni interval, v katerem je podatkovni strežnik preobremenjen?
- Kako odkriti poizvedbo na podatkovnem strežniku, ki povzroči preobremenitev?
- Kako pojasniti razloge, zaradi katerih poizvedba povzroči preobremenitev?

Performance Logs and Alerts se je izkazal kot učinkovito orodje za spremljanje zasedenosti sistemskih virov in obveščanje ob preobremenitvah podatkovnega strežnika. `Sp_trace` nudi podroben vpogled v izvajanje opravil, ki jih opravlja podatkovni strežnik. Uporabnike namreč najbolj zanimata hitrost izdelave poročil in učinkovitost servisiranja

zahtev, obe omenjeni orodji pa to izvajanje učinkovito spremljata. Načrt izvrševanja poizvedb sicer daje podroben opis posameznih korakov pri njihovem izvajanju, vendar me preseneča, da se spremembe izvorne kode v nekaterih primerih niso odražale na načrtu. Pri eni poizvedbi sem s pomočjo načrta odkril njeno problematično lastnost, ko je izvorna verzija uporabljala počasnejšo metodo iskanja po indeksu.

V nadaljevanju na kratko komentiram dobljene rezultate na vseh treh poizvedbah. Popravljeno verzijo poizvedbe kazalnik KPI bi dal na voljo uporabnikom, hkrati pa bi skrbnikom priporočil, da je 10% pohitritev dovolj velika, da ni potrebe po nadgradnji strojne opreme. Poizvedbo Prihodki na projektih za tekoče in prejšnje leto bi dal na voljo uporabnikom, hkrati pa bi skrbnikom priporočil nadgradnjo strojne opreme. Pohitritev ni velika, vendar bi se ta povečala v primeru spremembe določenih lastnosti podatkovnega skladišča, npr. zrnatosti. Poizvedbe Rezultat pri naročnikih med leti 2006 in 2010 ne bi dal na voljo uporabnikom, ker se izvaja počasneje kot izvorna verzija. Skrbnikom bi predlagal, da nadgradnja strojne opreme predstavlja edini način za razrešitev zmogljivostnih težav podatkovnega strežnika.

Predmet diplomske naloge se je skozi proces nastajanja vsebine izkazal za zanimivega. Idejo za modeliranje poslovnih procesov, pripadajočo implementacijo podatkovne baze in podatkovnega skladišča sem namreč dobil iz resničnega oglaševalskega podjetja. Opisani sistem se je tako izkazal za učinkovitega, saj je v kratkem času generiral kompleksna poslovna poročila iz velike množice podatkov. Predstavljena orodja so pri tem v pomoč pri ohranjanju kakovosti strežniških storitev tudi ob tistih spremembah informacijskega sistema, ki povečujejo izrabo sistemskih virov podatkovnega strežnika.

LITERATURA

[1] N. Zimic, M. Mraz, Temelji zmogljivosti računalniških sistemov, Založba FE in FRI, 2006.

[2] W. H. Inmon, Building the Datawarehouse, Wiley, 2005.

[3] R. Kibmall, J. Caserta, The Data Warehouse ETL Toolkit, Wiley, 2004.

[4] (junij 2010) B. Graziano, Examining SQL Server Trace Files.
url: http://www.sqlteam.com/article/examining-sql-server-trace-files

[5] (junij 2010) SQL Server 2005 Books Online: Buffer Management.
url: http://msdn.microsoft.com/en-us/library/aa337525(SQL.90).aspx

[6] G. Fritchey, Dissecting SQL Server Execution Plans, Simple-Talk Publishing, 2008.

[7] K. Delaney, Inside Microsoft SQL Server 2005: Query Tuning and Optimization, Microsoft Press, 2007.

[8] (junij 2010) SQL Server 2005 Books Online: Execution Plan Caching and Reuse.
url: http://msdn.microsoft.com/en-us/library/ms181055(SQL.90).aspx

[9] (junij 2010) D. Pinal, Index Seek Vs. Index Scan (Table Scan).
url: http://blog.sqlauthority.com/2007/03/30/sql-server-index-seek-vs-index-scan-table-scan