ŠIAULIŲ UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

INFORMATIKOS KATEDRA

Deividas Jakšta
Informatikos specialybės (mag.) II kurso studentas

SQL SERVERIŲ EFEKTYVUMO TYRIMAI VIDUTINIŲ DUOMENŲ BAZIŲ PROJEKTUOSE

MAGISTRO DARBAS

Darbo vadovas:
Doc. V. Sirius

Recenzentas:
Lekt. L. Kaklauskas

Šiauliai, 2006/2007 m.m.

Turinys

1. ĮVADAS

Išradus raštą 3000 metų prieš mūsų erą, atsirado galimybė užrašyti ir išsaugoti visą tuo metu aktualią informaciją. Visais laikais žmonija stengėsi užfiksuoti kaip galima daugiau įvykių, pastebėjimų, atradimų ir šiaip kitų faktų, kurie buvo reikšmingi ano meto žmonėms. Tai vyksta ir dabar, sparčiai vystantis mokslui bei technologijoms įvairiose veiklos srityse kilo poreikis saugoti, apdoroti ir analizuoti visą aktualią ir reikalingą informaciją. Taip atsirado **duomenų bazės** terminas – tai organizuotas (susistemintas, metodiškai sutvarkytas) duomenų rinkinys, kuriuo galima individualiai naudotis.

Dar iki XX a. vidurio įvairūs duomenys buvo saugomi, perduodami popieriaus lape, tačiau atsiradus kompiuterinėms technologijoms duomenys buvo pradėti saugoti pasitelkus jų galimybes. Tai gerokai supaprastino duomenų perdavimą, dalinimąsi, dauginimą, skleidimą ir panašiai. Tačiau intensyviai daugėjant duomenų įrašams, atsirado problema kaip juos patogiai ir lengvai atrinkti ir analizuoti. Tokiems procesams valdyti buvo sukurtos **duomenų bazių valdymo sistemos** (**DBVS**) (angl. **D**ata **B**ase **M**anagement **S**ystem **DBMS**). Naudojantis tokiomis sistemomis galima lengvai ne tik išsaugoti, atnaujinti bei peržiūrėti duomenis (informaciją), bet ir juos rūšiuoti, filtruoti bei grupuoti, kas palengvina duomenų atranką ir supaprastiną duomenų analizę.

Norint susikurti patogią ir efektyviai veikiančią **duomenų bazę** (**DB**), reikia įvertinti visus faktorius, kurie tiesiogiai įtakoja jos veikimą. **DB** efektyvumas priklauso nuo techninių sprendimų bei pasirenkamos programinės įrangos.

Tikslas:

1. Sukurti SQL serverių efektyvumo testavimo sistemą, kurioje nustačius tam tikrus parametrus, būdingus konkrečiai **DB**, atliktų testavimo darbus ir pateiktų testo rezultatus, kuriuos galima analizuoti ir įvertinti ar pasirinkta aparatinė įranga, operacinė sistema bei **DBVS** bus tinkama tai duomenų bazei valdyti ir aptarnauti.

Tam tikslui įgyvendinti yra keliami tokie **uždaviniai**:

1. suprojektuoti duomenų bazę ir aprašyti **SQL** kalba;
2. sugeneruoti meta duomenis ir užpildyti jais suprojektuotą duomenų bazę;
3. sugalvoti duomenis filtruojančias **SQL** užklausas (iš vienos ar kelių lentelių);
4. sistemos (atliekančios testavimą, rezultatų analizavimą) projektavimas ir sukūrimas.

2. TEORINĖ DALIS

2.1. Temos analizė

- **TPC** – **T**ransaction **P**rocessing **P**erformance **C**ouncil (oficialus puslapis - http://www.tpc.org/);
- **eWeek** – The Enterprise Newsweekly (oficialus puslapis - http://www.eweek.com/)

2.1.1. *TPC*

TPC – tai ne pelno siekianti organizacija, kuri įsikūrė 1985 metais. Jų tikslas atlikti testus su duomenų bazių valdymo sistemomis ir pateikti tikslius, objektyvius bei išsamius testo rezultatus.[6]

Tam tikslui **TPC** kompanija yra sukūrusi ne vieną testą, kurie turi skirtingas paskirtis. Dabar aktyviai naudojami testai:

- TPC-App
- TPC-C
- TPC-H

Yra ir tokių testų, kurie nebenaudojami, jų buvo atsisakyta arba jie yra integruoti į dabar naudojamus testus:

1. TPC-A
2. TPC-B
3. TPC-D
4. TPC-R
5. TPC-W

Dauguma pagrindinių informacinių technologijų organizacijų, kurios labai gerai žinomos visame pasaulyje, yra **TPC** organizacijos nariais. Nariai yra ne tik **DBVS** kūrėjai, bet ir stambiausios procesorių ir kompiuterių (serverių) gamintojai. Jie taip pat gauna testo rezultatus ir analizuoja, kaip jų sukurti procesoriai ar kompiuteriai sugeba dirbti su tam tikromis sistemomis prie didelių apkrovimų. Tai gi negalima teigti, kad **TPC** organizacijos veikla yra sufokusuota vien tik į

duomenų bazių valdymo sistemų testavimą, tuo pačiu ji pateikia informaciją, susijusią ir apie įrenginius su kuriais buvo atliekami testai.

2.1.1.1. TPC organizacijos nariai

AMD	BULL	DELL	FUJITSU	hp invent
HITACHI	IBM	INGRES	intel	Microsoft
Teradata a division of NCR	NEC	NETEZZA The Power to Question Everything	ORACLE	FUJITSU SIEMENS COMPUTERS
Sun microsystems	SYBASE	UNISYS		

2.1.1.2. Testo aprašymas

TPC kompanija viešai neplatina savo testų, kaip programos. O tai reiškia, kad vartotojai (įmonės, įstaigos) negali pasinaudoto **TPC** produktu ir pasižiūrėti kaip veikia jų serveris su konkrečia **DBVS**. Iš tiesu **T**ransaction **P**rocessing **P**erformance **C**ouncil atlieka labai didelį darbą, jie atlieka testus su specializuotais GRID, Cluster, Rack-Mountable, Blade tipo kompiuteriais. Pagal tokių testų publikacijas galima spręsti ar sistema, su kuria buvo atliekami testai, bus tinkama konkrečiam atvejui, tačiau tai labiausiai tinka tik didelėms ir labai didelėms DB. Vidutinių duomenų bazių kūrėjams netinka tokių testų aprašymai, kadangi yra labai didelių neatitikimų techniniuose sprendimuose. taigi nėra aišku, kokia sistema jiems tinka optimaliausiai veikimo ir kainos atžvilgiu.

TPC neplatina savo projekto, tačiau pateikia aprašymą apie jį, taip pat duomenų bazės specifikaciją (žiūrėti Priedai\TPCE-v1.0.0.pdf 42 psl.) ir visų lentelių aprašymus (žiūrėti Priedai\TPCE-v1.0.0.pdf nuo 43 psl. iki 57 psl.).

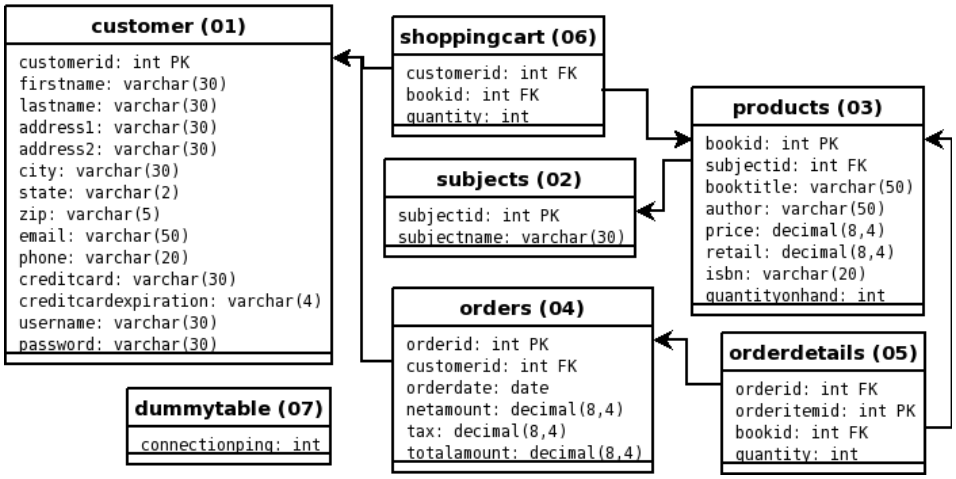
2.1.2. *„eWeek“*

„eWeek“ (Enterprise Newsweekly) – tai informacinių technologijų ir kompiuterinio verslo žurnalas, savaitraštis. Jame publikuojami straipsniai apie IT naujienas, testus, apžvalgas ir ateities planus. Šio žurnalo temos dažnai yra sudėtingos suvokti, taigi jo straipsniai yra rašomi daugiau IT

profesionalams nei mėgėjams. Šis žurnalas ankščiau buvo žinomas kaip „***PCweek***“. Ne išimtis, kad šis žurnalas nutarė parašyti straipsnį apie duomenų bazių valdymo sistemas ir jų efektyvumą. Tam, kad galėtų objektyviai įvertinti kelias sistemas ***eWeek*** taip pat sudarė testą, kurį vykdė **DBVS** ir fiksavo testo rezultatus.

eWeek priešingai nei **TPC** nesukūrė testų įvairioms operacinėms sistemoms. Sukurta testų sistema buvo skirta atlikti tik su **Microsoft Windows** šeimos operacinėmis sistemomis. ***eWeek*** savo testavimo sistemos visiškai neslepia nuo vartotojų. Jie gali pasinaudoti šiuo testu, kadangi jis yra laisvai platinamas ir nemokamas. Tačiau parsisiuntus testą norint jį išbandyti, reikia parašyti šio žurnalo redakcijai elektroninį laišką su prašymu, kad jie atsiustų duomenis, kuriais bus užpildyta **DB**. Redakcijos teigimu duomenys užima ≈1 GB vietos [7].

eWeek testo duomenų bazės schemą sudariau iš scenarijų (skriptų) failų (žiūrėti Priedai\eWeek\dbbenchmark_v1\database scripts).



1 diagrama

2.2. Darbo srities analizė

2.2.1. Duomenų bazių klasifikacija

Duomenų bazės yra skirstomos į:

- mažas DB (~10 000 arba > 1 GB);
- vidutines DB ($\leq 1*10^7$ arba < 10 GB);
- dideles DB ($\leq 1*10^9$ arba ≤ 1 TB);
- labai dideles DB ($>1*10^9$ arba 100 TB);
- duomenų saugyklos ($>1*10^9$ ir ∞);

Šiomis dienomis Lietuvos rinkoje dominuoja vidutinės duomenų bazės. **Vidutinės duomenų bazės** pagrindiniai parametrai:

- Duomenų bazėje yra $\leq 1*10^6$ įrašų arba ji užima ≤10 GB vietos;
- Vidutiniškai 10 000 užklausų per 1 valandą.

2.2.2. Reliacinės duomenų bazių valdymo sistemos

Reliacinės duomenų bazių valdymo sistemos - tai tokios **DBVS**, kurios remiasi Edgaro Kodo (*Edgar F. Codd*) aprašytu reliaciniu modeliu (1970 m.).

Formaliai reliacinio modelio pagrindinė sąvoka yra **lentelė**. Reliaciniame modelyje lentelę sudaro horizontalios **eilutės** (angl. row) arba **įrašai** (angl. record) ir vertikalūs **stulpeliai** (angl. column) arba **laukai** (angl. field). Kiekvienas laukas esantis lentelėje turi savo pavadinimą. Lauko duomenų tipas yra vienodas. Įrašas gali turėti kelis laukus ir jie nebūtinai turi būti vienodo tipo. Kiekvienas įrašas turi savo adresą, taip galime susieti vienos lentelės įrašus su kitos lentelės įrašais. Taigi reliacinę duomenų bazę sudaro **lentelių rinkinys**.

Dauguma šiuolaikinių populiariųjų **DBVS** palaiko didžiąją dalį E. Kodo suformuluotų taisyklių: *Oracle, Microsoft SQL Server, IBM DB2, MySQL, PostgreSQL*, ir kitos, tačiau jomis neapsiriboja.

2.2.3. Duomenų bazių valdymo sistemos funkcijos

Duomenų bazės valdymo sistema (**DBVS**) – vadinama programine įranga, skirta **DB** kurti, jas saugoti ir įvairiais būdais apdoroti. Svarbiausios **DBVS** funkcijos yra šios:

- **DB** struktūros projektavimas;
- **DB** pildymas, kaupimas, redagavimas;
- navigacija **DB**;
- duomenų peržiūra, paieška, rikiavimas ir kitas tvarkymas;
- taikomųjų vartotojo programų kūrimas;
- ataskaitų kūrimas.

Taip pat **DBVS** turi kur kas sudėtingesnius uždavinius:

- Naudojant popierinę duomenų saugojimo technologiją, norint tuos pačius duomenis naudoti sprendžiant skirtingus uždavinius, dažnai tenka duomenis dubliuoti. Pvz.: sandėlys turi savo prekių sąrašą, buhalterija - savo ir pan. Tokia situacija nėra pageidautina, todėl **DBVS** keliamas uždavinys **minimizuoti duomenų perteklių**;
- Dirbant su failais, vienam vartotojui atidarius failą rašymui kiti vartotojai negali jo pasiekti, ši situacija nėra pageidaujama, dėl to **DBVS** keliamas uždavinys yra užtikrinti **efektyvų bendrą DB naudojimą**;
- **DB** vadinama **vientisa (integrali)**, jei tenkina tam tikrus apribojimus (sąlygas) duomenims ir išsaugo tuos apribojimus modifikuojant (keičiant, šalinant, įterpiant) duomenis. Tokio apribojimo pavyzdys: „studento bakalauro kursas negali būti didesnis nei 4“;
- Labai svarbu apsaugoti duomenis nuo tyčinio ar netyčinio reikalingų duomenų sunaikinimo (pašalinimo) ar pakeitimo. Duomenų bazės **saugumo** sąvoka apima ir apsaugą nuo neleistinos domenų peržiūros. Šiam tikslui pasiekti, **DBVS** suteikia atsakingam asmeniui (administratoriui) priemones, leidžiančias apibrėži kiekvieno vartotojo ar jų grupės teises duomenų atžvilgiu.
- Saugant duomenis failuose, įmanoma sukurti programas, efektyviai atliekančias paiešką net ir labai dideliuose failuose pagal iš anksto numatytus kriterijus. Kadangi iš anksto numatyti visus galimus paieškos kriterijus ne visada įmanoma - **DBVS** keliamas uždavinys dideliuose duomenų masyvuose efektyviai atlikti ne tik **planuotas užklausas** (pagal iš anksto numatytus kriterijus), bet ir **neplanuotas užklausas**.

2.2.4. *SQL*

SQL (Struktūrizuota užklausų kalba, **Structured Query Language**) – tai gana paprasta kalba su aiškiomis standartizuotomis instrukcijomis, skirtomis aprašyti duomenis ir manipuliuoti jais reliacinių duomenų bazių valdymo sistemose.

SQL kalbą sudaro keletas sakinių grupių:

- **DDL** – (angl. Data Definition Language) duomenų apibrėžimo sakiniai,
- **DML** – (angl. Data Manipulation Language) manipuliavimo duomenimis sakiniai,
- **DCL** – (angl. Data Control Language) duomenų valdymo sakiniai.

2.2.4.1. Duomenų išrinkimas

ANSI bei ISO standartai apibrėžia šiuos **SQL** raktažodžius, skirtus duomenims išrinkti:

- `SELECT` komanda naudojama įrašams iš vienos ar daugiau lentelių atrinkti (šie įrašai dažniausiai atrenkami pagal tam tikrus kriterijus);
- `FROM` komanda nurodo lenteles, iš kurių reikia išrinkti eilutes (sąryšiai gali būti nurodomi skirtingais `JOIN` variantais);
- `WHERE` komanda nusako sąlygą, kurią turi tenkinti grąžinamos eilutės;
- `GROUP BY` komanda nurodo, įrašų grupavimo taisykles. Grupuojant eilutes, dažniausiai naudojamos agregatinės funkcijos maksimalioms, vidutinėms ir panašioms reikšmėms išrinkti iš grupuotų eilučių;
- `ORDER BY` komanda nurodo įrašų rikiavimo sąlygas;
- `HAVING` nurodomas kriterijus, taikomas grupuojamoms eilutėms; ši komanda gali būti naudojama tik tais atvejais, jeigu užklausoje yra `GROUP BY` komanda.

2.2.4.2. Duomenų valdymas

- `INSERT` vartojamas naujų įrašų įterpimui į lentelę;
- `DELETE` leidžia ištrinti įrašus iš lentelės;
- `UPDATE` naudojamas pakeisti vieno ar daugiau įrašų reikšmes.

2.2.4.3. Transakcijos

Sistemose, kurios palaikomos transakcijos, galima naudoti šias komandas:

- `BEGIN` nurodoma pradėti atominę operaciją (transakciją);
- `COMMIT` patvirtinama sėkmingai baigiama transakcija;
- `ROLLBACK` nurodoma, kad visa transakcija atšaukiama.

2.2.4.4. Duomenų apibrėžimas

- `CREATE` naudojama sukurti įvairiems objektams, pavyzdžiui, lentelėms;
- `DROP` nurodoma sunaikinti tam tikrus objektus.

Kai kuriuos sistemos turi komandą `ALTER`, kuria galima pakeisti objektus sistemos darbo metu.

2.3. Darbinės srities modelis

Duomenų bazių valdymo sistemų yra labai daug, visos jos yra skirtos duomenų bazių valdymui. Jos plačiai naudojamos įvairiose įstaigose, tačiau kiekvieną kartą projektuojant duomenų bazes ir norint jas pritaikyti yra sunku, kai reikia nuspręsti kokia **DBVS** yra tinkama ir patenkins konkrečios įstaigos poreikius su optimaliom išlaidom. Vienos jų yra komercinės, kitos nemokamos ir/ar atviro kodo, tačiau būtų klaidinga pasirinkti vien tik pagal jų kainą. Reikia atsižvelgti ir į pačią duomenų bazę - jos struktūrą, daugmaž numanomą duomenų kiekį, jų tipus, apytiksliai kiek vartotojų dirbs su šia duomenų baze vienu metu ir panašius kriterijus. Remiantis šiais kriterijais galima pradėti rinktis konkrečią sistemą, tačiau vėl iškyla dilema, kadangi visi šių sistemų kūrėjai reklamuoja, kad jų produktai yra patys geriausi. **DBVS** oficialiuose puslapiuose (*home page*) atsiranda dar daugiau neaiškumų, kadangi kiekvienas pateikia tik jiems naudingą informaciją. Taigi tikros tiesos apie norimą sistemą negauname, nes yra daug prieštaravimų. Daugiau būna aišku kai informaciją teikia tie šaltiniai, kurie nėra suinteresuoti parduoti kažkokį produktą, o dar geriau jei tie skleidėjai yra konkrečios ar/ir konkrečių sistemos ar/ir sistemų vartotojai. Jie gali pateikti savo pastebėjimus apie vienokią ar kitokią **DBVS** ir šie pranešimai būtų labiau objektyvesni. Tačiau ir čia mes turėtume pasitikėti kažkieno nuomone.

Taisyklingiausią ir išsamiausią informaciją apie duomenų bazių valdymo sistemas gali pateikti nepriklausomų tyrėjų grupės. Yra kelios kompanijos, kurios atlieka tokius tyrimus. Patys **DBVS** kūrėjai pristato dvi kompanijas, kurios atlieka, arba atliko nepriklausomus testus su jų produktais ir pateikė oficialius tų testų rezultatus:

- **TPC** – **T**ransaction **P**rocessing **P**erformance **C**ouncil;
- **eWeek** – The Enterprise Newsweekly.

Šių kompanijų testų privalumus ir trūkumus jau aptarėm **2.1.1** ir **2.1.2** skyriuose. Taigi įvertinus šių organizacijų atliktus darbus nusprendžiau sukurti dar vieną testavimo sistemą, kuri leis nustatyti **SQL** serverių efektyvumą.

Tokios sistemos privalumai turėtų būti:

- Daugiaplatformiškumas (Cross-platform) – sistema neturi būti skirta konkrečiai operacinei sistemai;
- Lengvas sistemos valdymas – vartotojo sąsaja turi būti suprantama, kad sistemą galima būtų valdyti intuityviai, be jokių ypatingų žinių ir pastangų;
- Parametrų keitimo galimybė – vartotojas gali keisti testo parametrus, kurie leis atlikti įvairesnius testus su pasirinktomis technologijomis (pvz.: galimybė pakeisti generuojamų duomenų kiekį – taip galime testavimo metu modeliuoti būsimos duomenų bazės dydį ir matyti veikimo rezultatus);
- Aiškus testo rezultatų pateikimas;

Tokia sistema padėtų pasirinkti duomenų bazių valdymo sistemą pagal kiekvieno poreikius kuriant vidutines duomenų bazes.

3. PROJEKTINĖ DALIS

3.1. Įrankių ir priemonių pasirinkimo analizė

3.1.1. Operacinių sistemų parinkimas

Kiekvienas IT specialistas administruojantis serverį, turi savo nuomonę apie tinkamiausią operacinę sistemą serveriams. Tos nuomonės dažnai skiriasi, kadangi yra pakankamai daug operacinių sistemų skirtų serverių valdymui (žiūrėti Priedai\II aprasymas.doc).

Taigi pagal jų klasifikavimą šiam tiriamajam darbui tinkamos sistemos priklauso trims operacinių sistemų šeimoms. Jos skirstomos į **Win32** šeimos OS (arba kitaip Windows), **Unix - Like** šeimos OS (arba kitaip Linux) ir **BSD** šeimos OS (arba kitaip Unix).

Pagal šį suskirstymą buvo pasirinkta po vieną operacinę sistemą:

- Windows 2003 server;
- Slackware 11.0;
- FreeBSD 5.4.

3.1.2. DBVS parinkimas

Renkantis duomenų bazių valdymo sistemas, buvo atsižvelgta į reliacinių duomenų bazių modelio palaikomumą. Taigi atrinktos šios duomenų bazių valdymo sistemos:

- IBM DB2 9.1.0.356
- Microsoft SQL Server 2005
- MySQL 5.0
- Oracle 10g
- PostgreSQL 8.2

Šios 5 sistemos buvo pasirinktos darbo atlikimui.

3.1.3. Kitų įrankių parinkimas

Parašyti programą, kuri turi sąryšį su **DB** galima įvairiausiomis programavimo kalbomis. Šiuo aspektu, renkantis programavimo kalbas, ypatingų apribojimų nėra. Tačiau mano iškeltas tikslas, kad kuriama sistema turi būti daugiaplatformė, iškarto sumažino pasirinkimų kiekį. Patogiausias tokios sistemos pavidalas yra internetinė sistema (*web aplication*), taigi pasirenkamos programavimo priemonės ir yra susijusios su internetinių sistemų kūrimu. Žinoma, kad **HTML** (**H**iper **T**ext **M**odeling **L**anguage) kalba yra būtina rašant web aplikacijas. Taip pat neišvengiama **PHP** programavimo kalba, kadangi per ją galima organizuoti prisijungimą prie visų mano pasirinktų **SQL** serverių ir juos valdyti. Tam, kad ši aplikacija būtų lengviau valdoma ir patraukli akiai, teks pasinaudoti **JavaScript** kalba. Ši kalba patogi tikrinant ar visi reikalingi formų laukai yra užpildyti ir panašiai.

3.2. Projekto (darbo) vykdymo planas

Projekto vykdymą galima suskirstyti į šiuos etapus:

1. Temos analizė (12 mėn.);
2. Projektavimas (3 mėn.);
3. Sistemos sukūrimas (1 mėn.);
4. Sistemos testavimas (2 mėn.);
5. Gautų rezultatų analizavimas (2 d.);
6. Projekto aprašymas (1 mėn.).

3.3. Pradinis projekto aprašymas

Projektuojant sistemą tiriančią **SQL** serverių efektyvumą, buvo sudarytas toks darbų planas:

- Suprojektuoti modelinę duomenų bazę, kuri bus naudojama tyrimo metu;
- Parašyti transakciją **SQL** kalba, kuri sukurs visas reikalingas lenteles pagal modelinės duomenų bazės projektą;
- Sugalvoti ir aprašyti **SQL** kalba duomenų užpildymo mechanizmą, kurio pagalba būtų nesunkiai užpildyta modelinė duomenų bazė meta duomenimis;

- Sugalvoti duomenis filtruojančias **SQL** užklausas iš vienos ar kelių lentelių (taip pat jas aprašyti), kurių vykdymo metu gautume jų eigos rezultatus ir galėtume juos analizuoti;
- Suprojektuoti bei suprogramuoti sistemą (programą), kuri įvykdytų visas transakcijas ir pateiktų tų transakcijų atlikimo laikus.

3.3.1. Modelinės duomenų bazės projektavimas

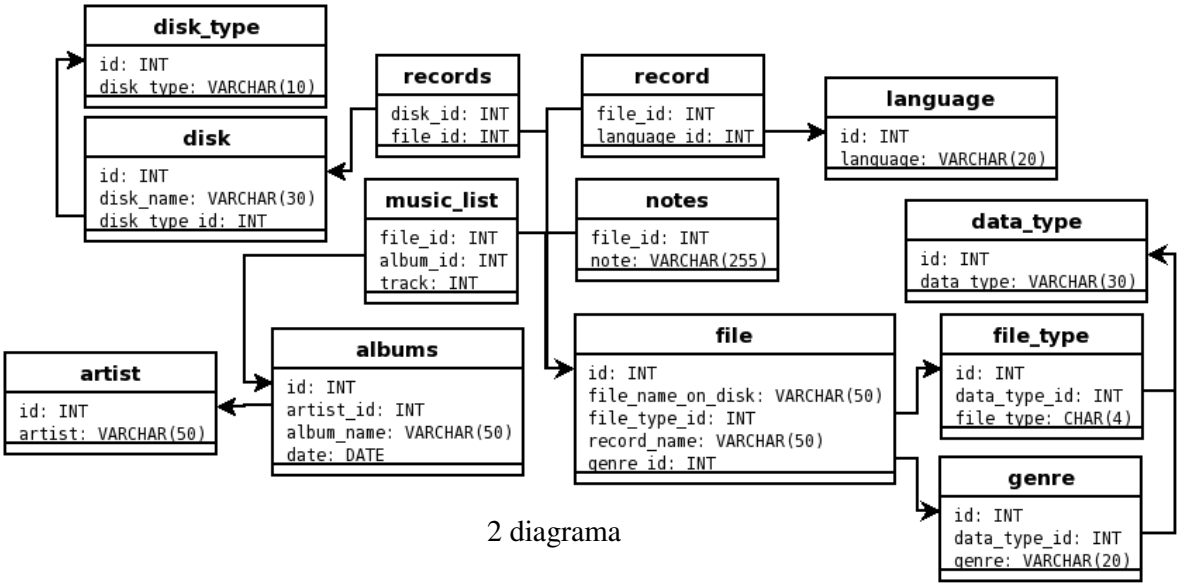
Kuriama testavimo sistema, turi įvertinti įvairiausias testuojamo objekto sąlygas. Šiuo atveju yra testuojamos duomenų bazių valdymo sistemos. Projektuojama modelinė duomenų bazė turi būti kuo panašesnė į įvairius duomenų bazių projektus. Taigi projektuojamai modelinei duomenų bazei buvo iškelti tokie reikalavimai:

- Duomenų bazėje turi būti mažų lentelių;
- Duomenų bazėje turi būti didelių lentelių;
- Lentelėse turi būti simbolių eilutę atitinkančių duomenų;
- Lentelėse turi būti skaitinių duomenų;
- Lentelėse turi būti loginių duomenų;
- Lentelėse turi būti datos arba laiko duomenų;
- Lentelėje turi būti binarinio kodo (failų) išsaugojimo galimybė;
- Simbolių eilutės turi būti mažos apimties;
- Simbolių eilutės turi būti vidutinės apimties;
- Simbolių eilutės turi būti didelės apimties.
- Modelinėje duomenų bazėje turi būti realizuotas ryšis vienas prie daugelio;
- Modelinėje duomenų bazėje turi būti realizuotas ryšis daugelis prie daugelio (pasinaudojus sąrašų lentelę).

Įvairiuose šaltiniuose (knygose, internetinėse svetainėse, pagalbos sistemose ar žinynuose) dažniausiai pasitaikanti pavyzdinė duomenų bazė yra susijusi su pirkėjais, pardavėjais, prekėmis ir ta tema aktualia informacija. Taigi projektuojant savo modelinę duomenų bazę stengiausi išvengti tipinių ir dažniausiai pasitaikančių projektų, kadangi jau šia tema yra pakankamai gerų pavyzdžių. Mano suprojektuotą duomenų bazę galima pavadinti „Failų kartoteka“. Čia yra saugoma informacija apie įvairaus tipo failus, pradedant nuo tekstinių ar foto nuotraukų failų baigiant video filmų ar muzikinių failų aprašymo. Tačiau tokiuose projektuose nėra svarbus duomenų bazės

pavadinimas ar paskirtis svarbiausia, kad duomenų bazė tenkintų anksčiau iškeltus reikalavimus ir būtų, kuo universalesnė ir panašesnė į kitus projektus. Juk **DBVS** visiškai nesvarbu koks duomenų bazės pavadinimas ir ką ta duomenų bazė saugo. **DBVS** skirtingai reaguoja į skirtingus duomenų tipus, jų ilgius, ir kaip aprašytas sąryšis tarp visų duomenų.

Suprojektuotos duomenų bazės schema:



2 diagrama

Ši schema ne visiškai atitinka iškeltus reikalavimus (žiūrėti **4.2.** skyrių).

3.3.2. Lentelių sukūrimas

Visos **3.1.2.** skyriuje išvardintos **DBVS** palaiko **SQL2** standartą, taigi remiantis šiuo standartu buvo aprašyta transakcija, kuri sukurs visas lenteles su jų laukais ir sąryšių aprašymais pagal sukurtą **DB** projektą. Transakcijos skriptas yra Priedai\tables.txt. Tačiau, kad ir visos sistemos palaiko **SQL2** standartą, bet tarp jų atsirado neatitikimų (žiūrėti **4.2.** skyrių).

3.3.3. Duomenų užpildymo mechanizmas

Norint pradėti atlikinėti testus su duomenų bazėmis, visu pirma reikia į jas įvesti reikiamą duomenų kiekį. Pagal mano iškeltus kriterijus testavimo sistemai (žiūrėti **2.3.** skyrių), generuojamą duomenų kiekį gali pasirinkti vartotojas, pagal savo nuožiūrą.

Duomenų kiekis lentelėse nevienodas. Bendras duomenų kiekis yra suskirstomas procentaliai:

- *disk_type* lentelę sudaro 2% visų duomenų;
- *data_type* – 3%;
- *language* – 5%;
- *file_type* – 5%;
- *genre* – 5%;
- *disk* – 10%;
- *artist* – 10%;
- *albums* – 15%;
- *files* – 45%;

Sąrašų lentelėse (lentelės, kuriose nėra pirminio rakto: *records*, *record*, *music_list*, *notes*) duomenų kiekis generuojamas atsitiktinai, neviršijant lentelės *files* įrašų kiekio, t.y. ne daugiau nei 45% pageidaujamo įrašų kiekio.

Iš pradžių buvo suprojektuotas duomenų generavimas atsitiktiniu būdu pasinaudojus **PHP** programavimo kalba ir juos surašant į failą kaip **SQL** sakinį (pvz.: *INSERT INTO artist VALUES (1,'rqNFxxTpLDvRnJC');*). Vėliau, visus sugeneruotus duomenis su įterpimo sakinio sintakse, paleisti ir taip būtų suvesti visi duomenys į duomenų bazę. Realizavus tokią sistemą paaiškėjo, jog duomenų generavimas ir įrašymas į failą, vėliau suformuotų įterpimo užklausų iš failo skaitymas (kurios įveda reikšmes į duomenų bazes), užtrunka pakankamai ilgai. Todėl šio duomenų užpildymo būdo buvo atsisakyta ir nuspęsta generuoti duomenis pačioje **DBVS** iškarto juos įrašant. Pasinaudojus Dekartinės sandaugos principu, bei **DBVS** galimybe įvesti daugelį eilučių vienu *INSERT* sakiniu buvo sudarytas skriptas leidžiantis generuoti duomenis į **DB** (žiūrėti Priedai\insert.txt). Norint generuoti duomenis tokiu būdu, iš pradžių reikia įvesti kelis pradinius duomenis į lenteles, iš kurių vėliau bus generuojami kiti įrašai. Pradinių įrašų kiekis apskaičiuojamas tokiu principu: kiekvienos lentelės įrašų kiekį žymime *X*. Tuomet reikia apskaičiuoti pradinių įrašų kiekį, iš kurių bus generuojami duomenys: $\sqrt[8]{X} = y$. Iš gauto rezultato *y*

paimamas neapvalintas sveikasis skaičius ir šis skaičius nurodo pradinių duomenų kiekį lentelėse. Pradiniai duomenys generuojami anksčiau nepasitvirtinusiu principu, šiuo atveju jis yra tinkamas kadangi reikia sugeneruoti nedaug duomenų.

Sugeneruoti pradiniai duomenys yra suvedami į duomenų bazę, tuomet pasinaudojant jais ir ankščiau paminėtu generavimo principu (pačios **DBVS** viduje, pasinaudojant Dekarto sandauga ir daugelio eilučių įterpimo galimybe) generuojami visi duomenys.

3.3.4. Duomenų atrankos komandos

Duomenų atrankos komandos buvo kuriamos atsižvelgiant į šiuos apribojimus:

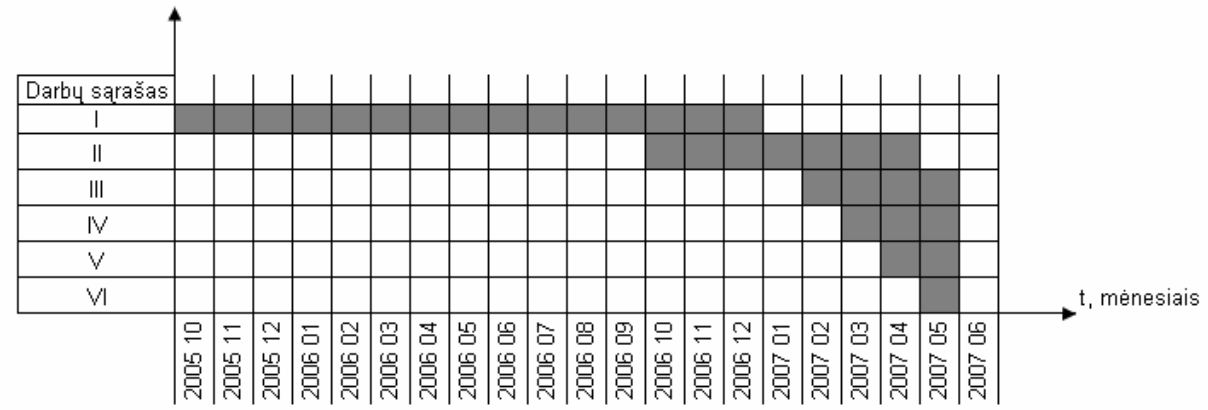
- kiekviena lentelė turi bent po vieną duomenų atrankos komandą;
- duomenys atrenkami iš dviejų ir daugiau lentelių, pagal jų sąryšius;

Viso sugalvota 15 duomenų atrankos komandų.

4. DARBO EIGOS APRAŠYMAS

4.1. Darbų eigos grafas

I. Temos analizė (2005-10/2006-09);
II. Projektavimas (2006-10/2007-04);
III. Sistemos sukūrimas (2007-02/2007-05);
IV. Sistemos testavimas (2007-03/2007-05);
V. **SQL** serverių efektyvumo tyrimas (2007-04/2007-05);
VI. Projekto aprašymas (2007-05).



4.2. Problemų ir jų sprendimų aprašymai ir pagrindimai

Atliekant šį darbą pirmosios problemos atsirado projektavimo dalyje. Projektuojant testavimo sistemą, o tiksliau modelinę **DB** buvo sudaryti kriterijai, kuriuos turi tenkinti suprojektuota **DB** (žiūrėti **3.3.1.** skyrių). Visus išvardintus kriterijus pavyko įgyvendinti išskyrus vieną, kuris skamba taip „*Lentelėje turi būti binarinio kodo (failų) išsaugojimo galimybė;*“. Suprojektuoti duomenų bazę kurioje galima saugoti failus taip pat įrašyti duomenis į tokią lentelę nėra sudėtinga. Tačiau **3.3.3.** skyriuje aprašytas duomenų generavimo ir užpildymo mechanizmas neturi galimybės atlikti tokių veiksmų su failais, kokie yra atliekami su skaitiniais, tekstiniais ar kito tipo duomenimis.

Viena iš rimtesnių problemų kilo, kai teko rašyti **SQL** transakcijas (užklausas). Visos šiame darbe naudojamos **DBVS** palaiko **SQL2** standartą, tačiau jos turi savo **SQL** dialektą, dėl to negalima buvo sudaryti vieno skripto (scenarijų) failo, kuris būtų tikęs visiems testuojamiems SQL serveriams. Teko kiekvienai **DBVS** sudaryti po skirtingą scenarijų failą atsižvelgiant į šių sistemų **SQL** kalbos dialektus.

Kitos problemos atsirado testavimo metu. Jos susijusios su sistemos veikimo laiku. Teko atlikti kai kuriuos pakeitimus (skaityti **3.3.3.** skyrių), kurių dėka buvo gerokai sutrumpintas testo veikimo laikas.

4.3. Galutinio projekto stovio aprašymas

Šio darbo metu buvo sukurta modelinė duomenų bazė. Sukurti scenarijų failai pagal kiekvieną **DBVS**, kurie:

- sukuria lenteles į duomenų bazę;
- užpildo jas pradiniais duomenimis;
- iš kurių yra generuojami reikiamas kiekis duomenų;
- atliekami duomenų keitimo sakiniai (UPDATE);
- vykdomi duomenų atrankos sakiniai (SELECT);
- ištrinamos lentelės;

Sukurta web aplikacija, į kurią yra sujungti visi scenarijų failai. Web aplikacija užfiksuoja kiekvienos transakcijos veikimo laiką ir įrašo į ataskaitų failus.

4.3.1 SQL serverių efektyvumo tyrimas

SQL serverių efektyvumo tyrimas buvo atliekamas su visomis **DBVS** įtrauktomis į testavimo sistemą, tai:

- IBM DB2 9.1.0.356
- Microsoft SQL Server 2005
- MySQL 5.0
- Oracle 10g
- PostgreSQL 8.2

Su visomis **DBVS** testai buvo atliekami vienodomis sąlygomis. Visos šios **DBVS** buvo suinstaliuotos į vieną ir tą patį kompiuterį.

Aparatinė įranga:

- Intel Pentium M Processor 730 1.60 GHz.
- 512 MB operatyvioji atmintis.
- Samsung MP0804h 80 GB kietasis diskas.

Operacinė sistema:

- Microsoft Windows XP Profesional su SP2 (antras pataisymų paketas).

Testas buvo atliekamas 30 kartų su kiekviena duomenų bazių valdymo sistema. Testo metu naudojami tie patys pradiniai duomenys. Taip pat visoms **DBVS** generuojami duomenys buvo pagal tas pačias taisykles. Iš viso buvo sugeneruota 1 032 900 duomenų visoje duomenų bazėje.

Testo rezultatai pateikiami laiko atžvilgiu, t.y. fiksuojamas kiekvienos transakcijos atlikimo laikas. Užfiksuotas laikas yra įrašomas į tekstinius failus. Norint matyti aiškius ir suprantamus rezultatus, o vėliau atlikti kokias nors išvadas, šiuos duomenis reikia apdoroti. Juos galima apdoroti įvairiais būdais. Aš įvertinau kiekvienos transakcijos vidutinį atlikimo laiką.

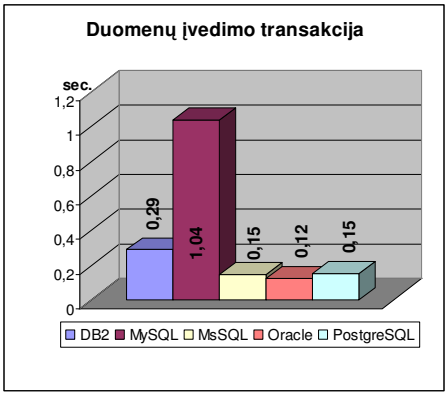
Viso testo rezultatai ir duomenys pateikti Priedai\testo_rez.xls. Aprašyme pateiksiu tik kelių (pačių reikalingiausių) transakcijų rezultatus:

- duomenų įvedimas;
- duomenų generavimas;
- duomenų keitimas;
- duomenų atranka.

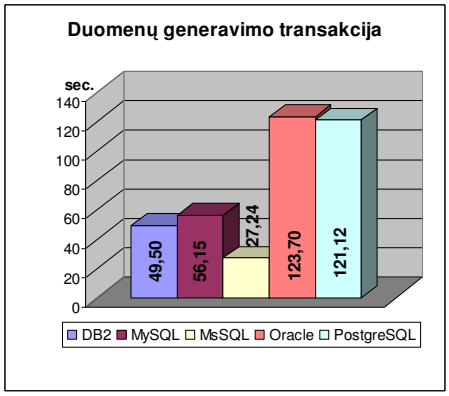
Šios funkcijos yra naudojamos kiekvienoje duomenų bazėje. Jos yra pagrindinės norint valdyti duomenų bazę. Kitos transakcijos, tokios kaip lentelių sukūrimas bei lentelių trynimas, nesudaro labai didelės svarbos duomenų bazės veikimui. Paprastai lentelės sukuriamos vieną kartą, ir retai kada jos yra trinamos.

Norint pasirinkti konkrečią duomenų bazių valdymo sistemą, visų pirma reikia išanalizuoti konkrečiai duomenų bazei iškeltus reikalavimus. Nuo konkrečios duomenų bazės paskirties galime sužinoti kokių transakcijų veikimo laikas yra svarbus, ir kad jis būtų kuo trumpesnis. Pavyzdžiui, mokslininkai atlieka tyrimus, kurių metu duomenys automatiškai suvedami į duomenų bazę. Viskas

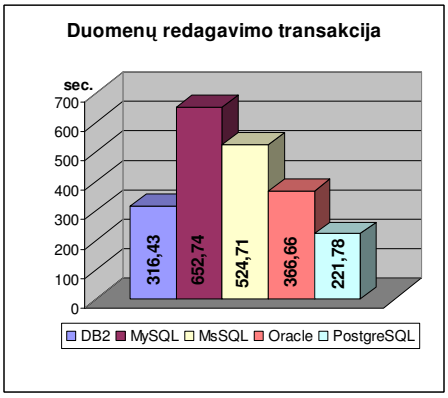
vyksta realiu laiku, taigi yra svarbu visus duomenis išsaugoti. Jei tų duomenų yra pakankamai daug, tai būtina užtikrinti greitą duomenų įvedimą. Tokiose duomenų bazėse labai greitas duomenų filtravimas nėra labai svarbus. Taigi pagal šį pavyzdį apribojimas yra greitas duomenų įvedimas. 3-čia diagrama, kaip tik ir parodo, kuri **DBVS** sugeba greičiausiai įvesti duomenis.



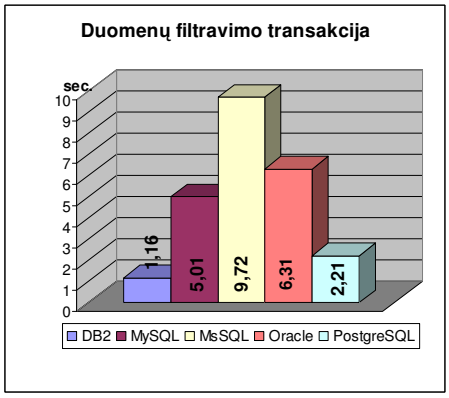
3 diagrama



4 diagrama



5 diagrama



6 diagrama

Šiame pavyzdyje buvo iškeltas tik vienas apribojimas (neskaitant duomenų kiekio). Kartais gali tekti iškelti kelis reikalavimus. Pavyzdžiui konkrečiai duomenų bazei reikia užtikrinti greitą duomenų filtravimą ir greitą duomenų redagavimą, tuomet reikia žiūrėti į dvi diagramas: 5-ą diagramą ir 6-ą diagramą. Jei šių reikalavimų svarbumas yra vienodas tuomet pakanka sudėti vidutinius šių transakcijų laikus atsižvelgiant į **DBVS** ir pasirinkti mažiausią laiką turinčią duomenų bazių valdymo sistemą. Jei svarbumas nevienodas, tuomet įsivedam koeficientą, kuris nurodys transakcijos svarbą. Taigi tarkime, kad projektuojamoje duomenų bazėje yra svarbus duomenų filtravimas 80%, o redagavimui belieka 20%. Tuomet vidutinius transakcijų laikus padauginame atitinkamai iš 0.8 ir 0.2 koeficientų ir sudedame gautą rezultatą. Taip pat mažiausiai užimantis laikas yra efektyviausias.

Gavus tam tikras išvadas negalime teigti jog konkreti duomenų bazių valdymo sistema tinkama visais atvejais. Kiekvienu atveju testo rezultatai gali skirtis. Viskas priklauso nuo:

- aparatinės įrangos;
- operacinės sistemos;
- apribojimų iškeltų konkrečiai duomenų bazei;
  - duomenų kiekis;
  - transakcijų svarba (jei reikia su koeficientais).

4.4. Darbo rezultatų analizė

Buvo planuota padaryti:

- Temos analizė;
- Projektavimas;
- Sistemos sukūrimas;
- Sistemos testavimas;
- **SQL** serverių efektyvumo tyrimas.

Įvykdyta:

- Temos analizė (2.1. Temos analizė);
- Projektavimas (3.3. Pradinis projekto aprašymas);
- Sistemos sukūrimas (Priedai\DBMStestsys);
- Sistemos testavimas (4.2. Problemų ir jų sprendimų aprašymai ir pagrindimai);
- **SQL** serverių efektyvumo tyrimas.

5. IŠVADOS

Šiuo metu dažniausiai sutinkamos duomenų bazės priskiriamos prie vidutinių **DB** klasės. Technologijoms besivystant, yra didesnė tikimybė pasirinkti gerą ir patikimą sistemą, tačiau tai atitinkamai kainuoja.

Jei diegiant vidutines duomenų bazes buvo pasirinkta netinkama aparatinė, operacinė bei programinė įrangos, vėliau tenka kažką keisti, kadangi **SQL** serveris neveikia taip, kaip buvo tikėtasi projektavimo metu, o tai vėl reikalauja išlaidų. Naudojantis sukurta **DBVS** testavimo sistema galime pamatyti ar prie tam tikrų parametrų bus tenkinamas vartotojų naudojimasis duomenų baze. Tokiu būdu duomenų bazių kūrėjams ar įstaigoms, norinčioms savo duomenis saugoti duomenų bazėje, bus lengviau nuspręsti kokia **DBVS** yra tinkamiausia, kad jų planuojama **DB** veiktų sklandžiai.

Atliekant šį darbą buvo:

- susipažinta su panašiais projektais;
- suprojektuota modelinė **DB**;
- Sugalvota duomenų generavimo sistema;
- Sukurta sistema atliekanti **DBVS** testavimą ir pateikianti testo rezultatus;
- Atliktas testas su penkiomis **DBVS**;
- Pateikti apdoroti tyrimo rezultatai.

6. LITERATŪRA

1. Oracle www.oracle.com
2. DB2 Product family http://www-306.ibm.com/software/data/db2/
3. Mycrosoft SQL Server http://www.microsoft.com/sql/default.mspx
4. MySQL http://www.mysql.com
5. PostgeSQL http://www.postgresql.org
6. PostgeSQL documentation http://www.postgresql.cl
7. WikipediA http://www.wikipedia.org
8. Transaction Processing Performance Council http://www.tpc.org
9. eWeek http://www.eweek.com
10. php http://www.php.net

7. KOMPAKTINIO DISKO TURINYS

\Aprasymas\

\Aprasymas\Deividas Jaksta.doc

\Aprasymas\Deividas Jaksta.pdf

\DBMStestsys\

\DBMStestsys\temp\

\DBMStestsys\temp\db2.txt

\DBMStestsys\temp\duomenys.txt

\DBMStestsys\temp\mysql.txt

\DBMStestsys\temp\mssql.txt

\DBMStestsys\temp\oracle.txt

\DBMStestsys\temp\pgssql.txt

\DBMStestsys\ajax.js

\DBMStestsys\db2.php

\DBMStestsys\generate.php

\DBMStestsys\index.php

\DBMStestsys\loading.gif

\DBMStestsys\mysql.php

\DBMStestsys\mssql.php

\DBMStestsys\oracle.php

\DBMStestsys\pgsql.php

\DBMStestsys\readme.txt

\Priedai\

\Priedai\eWeek\

\Priedai\eWeek\dbbenchmark_v1.zip

\Priedai\II aprasymas.doc

\Priedai\insert.txt

\Priedai\tables.txt

\Priedai\testo_rez.xsl

\Priedai\TPCE-v1.0.0.pdf

8. ANOTACIJA

Deividas Jakšta

SQL SERVERIŲ EFEKTYVUMO TYRIMAI VIDUTINIŲ DUOMENŲ BAZIŲ PROJEKTUOSE

Atliekant duomenų bazės projektavimo darbus visuomet yra svarbu žinoti ar pasirinkta sistema veiks taip kaip tikimasi. Šio darbo tikslas sukurti SQL serverių testavimo įrankį, kuris padės renkantis vienokią ar kitokią sistemą. Darbe išanalizuoti panašūs projektai, įvertinti tų projektų privalumai bei trūkumai. Sukurta sistema, aprašytas sistemos projektas ir veikimo principai.

9. ANNOTATION

Deividas Jakšta

THE SQL SERVER EFFICIENCY IN MEDIAL DATA BASE PROJECTS

Making a data base project always important to know - do the system will work properly. The main point of this job is to create the test system of SQL servers (or DBMS – Data Bases Management System) which will help to chose that or either machine configuration. In this work were analyzed the similar projects, appraised the necessity and imperfections of this projects.

Was created the system, written this system project and how it woks.