KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

INFORMACIJOS SISTEMŲ KATEDRA

Raigedas Radišauskas

Hierarchinių struktūrų reliacinėse duomenų bazėse saugojimo ir panaudojimo metodų tyrimas

Magistro darbas

Darbo vadovas
prof. Rimantas Butleris

KAUNAS, 2007

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Raigedas Radišauskas

Hierarchinių struktūrų reliacinėse duomenų bazėse saugojimo ir panaudojimo metodų tyrimas

Magistro darbas

Vadovas
prof. Rimantas Butleris
2007-05-28

Recenzentas
doc. dr. Antanas Lenkevičius
2007-05-28

Atliko
IFM-1/2 gr. stud.
Raigedas Radišauskas
2007-05-28

KAUNAS, 2007

Turinys

Lentelių turinys

Paveikslėlių turinys

Summary

Research of storage and retrieval methods in relational databases of the hierarchical data structures

Hierarchy (hierarchical tree) is a special graphical constract that fulfils the following conditions:

- Absence of cycles in the structure;
- Any of two nods are connected only on one path;
- A number of edges are lesser by one unit than the number of nods.

Hierarchical structures are encountered in a lot of areas of human activities. For example, in family genealogy to make a detailed picture of cognate relations among different family members, geneologists design a family tree with its stirps; in publishing to make it easier to find appropriate information in a piece of work, publishers create a table of contens with its headings and subheadings; or even human thinking proceeds on basis of hierarchical sturcture when the mind generates abstracts which are arranged in a hierarchical manner, enabling a person to remember things better (Boyle M.H., Willms J.D., 2001). Similarly, hierarchical trees are widely used in almost all fields of Computer Science, e.g. framing a operaing sistem, programers make up a file system with its file arrangements to easy the user's work with data on the computer.

The object of the work is to compare two methods that allow to store hierarchical structures in DBMS, and to evaluate their characteristics by experiment.

To achieve the object, the following objectives have been realized:

1. The comparison of some methods which enable to store trees in RBDVS has been made;

2. The reusable components which work on a number of different methods have been designed and implimented;

3. The experiment has been carried out.

1 Įvadas

Hierarchijos

Hierarchija (arba medis) yra specialus grafo atvejis, kai tenkinamos tokios sąlygos:

- nėra kilpų;
- bet kurios dvi viršūnės yra surištos tik vienu keliu;
- medžio briaunų skaičius yra vienetu mažesnis už viršūnių skaičių.

Problemos aktualumas

Su hierarchinėmis struktūromis mes susiduriame nuolat. Pradedant nuo genealoginio medžio, einant per failų sistemas, ir baigiant forumų antraščių medžiu. Magistro projektiniame darbe taip pat buvo realizuotos dvi medžio struktūros. Galų gale, šio darbo turinys yra taip pat medžio struktūra.

Deja, jei nenaudojama XML tipo duombazė, tai visi duomenys lentelėse nėra hierarchiniai; tai paprasčiausi plokšti sąrašai. Norint saugoti medžio struktūrą į RDBVS reikia surasti būdą, kaip konvertuoti hierarchijas į plokščius sąrašus.

Sprendimai

Darbo tikslas yra palyginti metodus, leidžiančius saugoti hierarchines struktūras reliacinėse DBVS, eksperimentiškai įvertinti jų charakteristikas. Tam reikia išspresti tokius uždavinius:

- palyginti metodus, leidžiančius saugoti medžius į RBDVS;
- suprojektuoti ir realizuoti pakartotino panaudojimo komponentus, dirbančius skirtingais metodais;
- atlikti eksperimentą.

2 Hierarchinių struktūrų būtinumo ir panaudojimo analizė

2.1 Hierarchinių struktūrų egzistavimas

Su hierarchinėmis struktūromis mes susiduriame nuolat. Pradedant nuo genealoginio medžio, einant per failų sistemas, ir baigiant forumų antraščių medžiu. Magistro projektiniame darbe taip pat buvo realizuotos dvi medžio struktūros. Galų gale, šio darbo turinys yra taip pat medžio struktūra.

Dar viena sritis kur dažnai naudojamos hierarchijos – žmogaus mąstymas, mintys. Žmogaus mastymo procesas gali būti pavadintas modeliavimu, abstrakcijų kūrimu ir lyginimu. O tos abstrakcijos rišamos į medžio struktūras, kas leidžia geriau įsiminti informaciją. Michael H. Boyle ir J. Douglas Willms savo darbe "Multilevel Modelling of Hierarchical Data in Developmental Studies" įrodė, jog hierarchinių struktūrų tikslingas naudojimas gali turėti teigiamų rezultatų vaiko vystymuisi [9].

Magistro projekte taip pat buvo reikalinga saugoti medžio struktūras. Jos buvo naudojamos dviejose posistemėse:

- išsaugant disko katalogų ir failų struktūrą;
- diskų aibę leidžiant sugrupuoti pagal norimus kriterijus, priskiriant juos grupėms, kurios organizuotos medžiu.

2.1.1 Failų sistemos

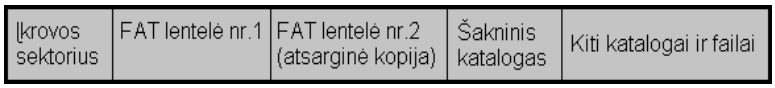
Beveik visos failų sistemos yra hierarchinės. Toliau apžvelgsime kelias iš jų.

2.1.1.1 FAT

FAT (angl. "File Allocation Table") yra vienos paprasčiausių failų sistemų, gerai demonstruojančių abstrakciją, kuri naudojama, failus išsaugant diske ar kitame įrenginyje. Paprasčiausia FAT sistema sudaryta iš trijų pagrindinių dalių, esančių griežtai apibrėžtose disko vietose [5]:

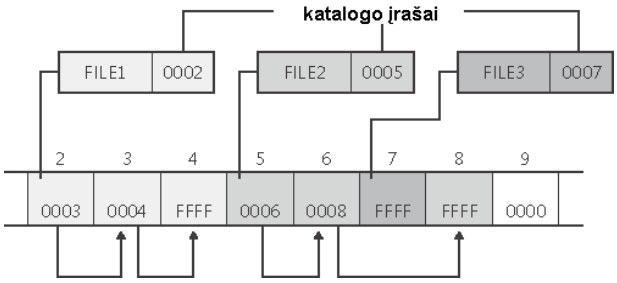
- failų alokavimo lentelės, kurioje yra visų klasterių sąrašas;
- šakninio katalogo, kuriame laikomas failų sąrašas;
- laisvos vietos, padalintos į apibrėžto ilgio (pvz., 512 baitų) klasterius.

FAT struktūra [13]:



1 Pav. FAT struktūra

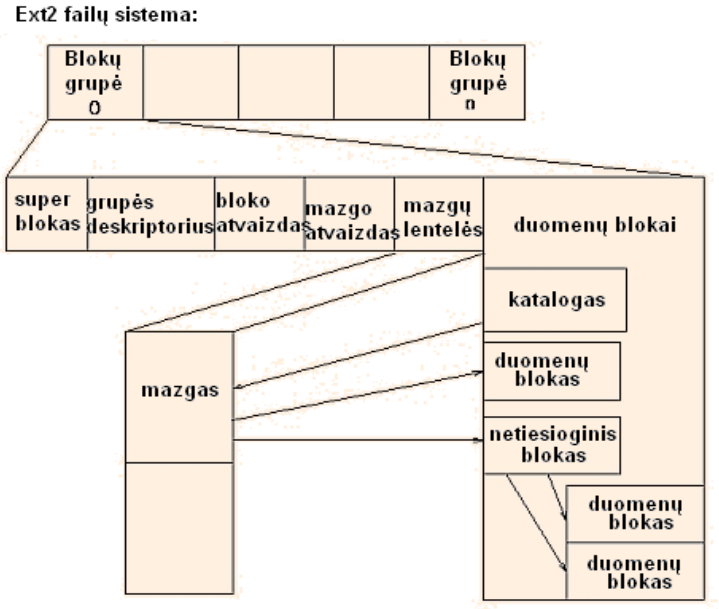
Kaip žinome, failų sistemose viršūnes atitinka katalogai. Katalogas realizuotas kaip eilinis failas, bet šiuo atveju vietoj failo turinio, turime to katalogo turinį. Pavyzdžiui, katalogo struktūra gali būti tokia, kokia pateikta 1 paveikslėlyje:



2 Pav. Pavyzdinė FAT grandinė

2.1.1.2 Ext2/Ext3

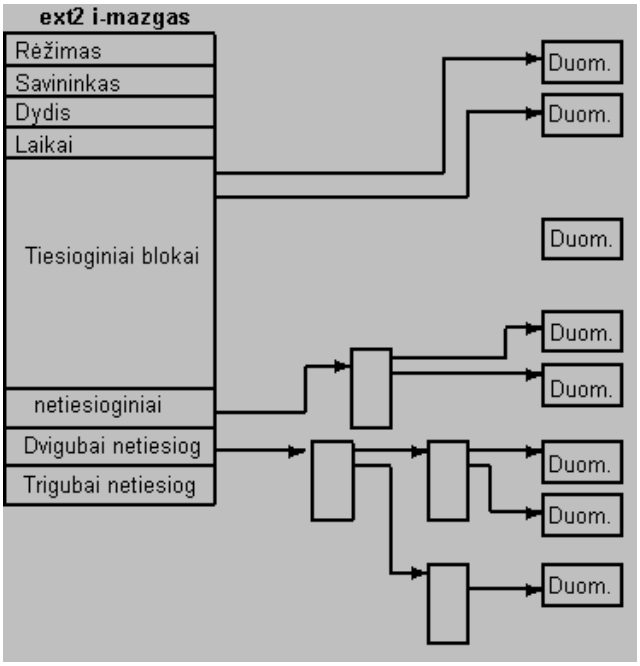
Ext failų sistema vietą paskirsto failų blokais, kurie organizuoti į blokų grupes [11], kaip parodyta 4 paveikslėlyje.



3 Pav. Ext2 failų sistema

Kiekvienas Ext failų sistemos failas turi su juo susietą vieną i-viršūnę (angl. "i-node"). Toje viršūnėje saugoma tokia informacija: failo savininkas, teisės, dydis, laikai (kūrimo, redagavimo, skaitymo) ir blokai, priskirti failui [12].

Ext failų sistemoje (priešingai nei FAT) ne tik katalogai bet ir fizinis failų išdėstymas sudaro medžio struktūrą. Rodyklės į duomenų blokus yra organizuotos medžiu (žr. 5 paveikslėlį). Failo medžio struktūros pažeidimai duoda rezultatą – prarandami blokai, esantys žemiau pažeisto bloko.



4 Pav. Ext2 i-viršūnė

Katalogas paprasčiausiai yra failas, kuriame surašyti jame esančių failų vardai ir jų i-viršūnių numeriai.

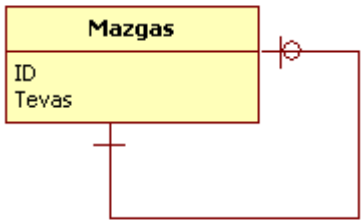
2.2 Hierarchinių struktūrų saugojimas reliacinėse duomenų bazėse

Atsiradus reliacinėms duomenų bazėms, atsirado ir poreikis juose išsaugoti medžio tipo duomenų struktūras. Toliau išnagrinėsime dažniausiai naudojamus būdus, leidžiančius tai atlikti.

2.2.1 Vaiko-tėvo modelis

Vaiko-tėvo modelis kituose šaltiniuose dar vadinamas gretimumų sąrašo modeliu (angl. "the

adjacency list model") [8]. Šį modelį pirmasis pasiūlė Edgar F. Codd [10]. Oracle buvo pirmo komercinė duombazė kuri panaudojo šį modelį. Šis modelis turbūt vienintelis, kurį galima aprašyti reliacine algebra. Pagrindinis šio metodo principas yra tas, kad kiekvienas viršūnės įrašas saugo ir savo tėvo identifikatorių. Kurdami ER modelį, nekreipsime dėmesio į dalykinės srities reikalavimus medžiui – laikysime, kad reikalingi duomenys surišami panaudojus viršūnės pirminį raktą ID.



5 Pav. Vaiko-tėvo modelis, ER

Pavaizdavimui naudosime tokį pavyzdinį medį:

1 Lentelė: Pavyzdinis medis

ID	*Parent*	*Name*
1	NULL	Elektronika
2	1	Televizoriai
3	2	Įprastiniai TV
4	2	LCD TV
5	2	Plazminiai TV
6	1	Nešiojama elektronika
7	6	MP3 grotuvai
8	7	Flash
9	6	CD grotuvai
10	6	"2 Way" radijo imtuvai

2.2.1.1 Įrašymas

Pagrindinis šio metodo privalumas yra įrašymo paprastumas ir greitis – norint sukurti naują viršūnę pakanka įterpti vienintelį įrašą, kuriame bus nurodyta, kas yra viršūnės tėvas.

2.2.1.2 Viršūnės trynimas

Viršūnė, neturinti vaikų, gali būti paprastai pašalinta, ištrynus vieną įrašą.

Viršūnės, kuri turi vaikų, negalima paprastai pašalinta nes tada medis skils į dvi dalis. Tuo atveju galimi keli pasirinkimai:

- visus vaikus pašalinti kartu;
- visiems vaikams suteikti naują tėvą – šalinamos viršūnės tėvą;
- vieną iš vaikų paversti tėvu.

2.2.1.3 Vaizdavimas – rekursija

Jau seniai informatikos moksle, rekursija laikoma elegantiškiausiu būdu pereiti žemyn visa duomenų hierarchija.

Svarbiausia priežastis, dėl kurios tai laikoma elegantišku sprendimu, yra ta, kad reikia parašyti tik vieną funkciją, apdorojančią, atvaizduojančią duomenis ir vykdančią iteracijas gilyn visa medžio struktūra. Pakanka tik vieną kartą iškviesti funkciją per parametrą paduodant šakninį medžio elementą ir viskas. Papildomas privalumas yra tas, kad tai galioja ne tik visam medžiui, bet ir jo poaibiui: pakanka funkcijai perduodi ne šakninį o bet kurį kitą elementą ir bus atvaizduotas visas medžio poaibis be papildomo logikos ir kodo keitimo. Pseudo-kodas, kuris atvaizduoja medį, galėtų būti toks:

```
function DisplayChildren(uid, indent_level)
  SELECT id, (info)
  FROM (db)
  WHERE ParentID = uid
  if recordcount > 0 then
    (indent by indent_level) (HTML to display info for this child)
    child_id = recordset.id
    DisplayChildren(child_id, indent_level+1)
  end if
end function
```

Trūkumai

Metodo trūkumas yra tas, kad didžioji dauguma programavimo kalbų turi labai prastai realizuotą rekursijos palaikymą. Esmė ta, kad kiekviena funkcija yra vykdoma savo atskiroje adresų erdvėje. Po to, kai funkcijos vykdymas baigiasi, atmintis turi būti atlaisvinta. Dėl to funkcijos kvietimas ir grįžimas iš jos yra pakankamai reiklūs resursams. Todėl pabandysime paskaičiuoti efektyvumą.

Pavyzdžiui, sakykime, kad turime tokias salygas:

- kiekvieno įrašo gavimas iš DB trunka n sekundžių;
- kiekvienas prisijungimas/atsijungimas prie DB trunka n * 2 sekundžių;
- atminties išskyrimas kiekvienai funkcijos kopijai trunka n/10 * iteracija sekundžių (rekursyvūs algoritmai kalbose "ASP", "Cold Fusion" turi logaritminį spartos mažėjimą priklausomai nuo iteracijos eilės).

Taigi, jeigu diskusijų forume būtų 1000 žinučių, lygtis būtų tokia:
(1000 * n) + (1000 * n * 2) + (n/10 * iteracija)
Geresniam vaizdui sudarome preliminarią lentelę:

2 Lentelė: Įrašo ištraukimo laikas

Įrašai	*Laikas sekundėmis*	*Eilė*
1	3 * n	1
250	29 * n	2
500	53 * n	3
750	79 * n	4
1000	103 * n	5

Suma = maždaug 50 000 * n sekundžių

Kaip matome, logaritminis spartos mažėjimas įpatingai pastebimas medžiuose, turinčiuose didesnį lygių skaičių.

Nors rekursija tai elegantiškas, lengvai suprantamas, bei kodu lengvai užrašomas būdas, bet, jei naudojama sistema nėra specialiai optimizuota rekursijai (optimizuotos sistemos pavyzdys būtų LISP), tai pridėtinės laiko sanaudos bus įpatingai didelės priklausomai nuo medžio lygių skaičiaus.

2.2.1.4 Vaizdavimas – stekas

Stekas yra antras būdas darbui su dideliomis medžių struktūromis. Yra du steko tipai: "First In First Out" ir "First In Last Out". Darbui su hierarchiniais sąrašais reikia naudoti FILO. Idėja yra ta, kad kurnors atmintyje sukuriamas stekas, kuris yra paprasčiausias sąrašas. Tada į steką įdedamas einamojo įrašo ID, o kai baigiamas įrašo apdorojimas, tuomet jo ID išimti iš steko. Jeigu tas įrašas turi vaikų, tai nuskaitomas pirmas vaikas bei įdedamas jo ID į steką ir t.t.

Bet tada kyla klausimas – kur sukurti ir dirbti su steku: taikomosios programos ar duomenų

bazės lygmenyje, t.y. SQL procedūroje (angl. "Stored Procedure").

Privalumai dirbant su steku taikomojoje programoje:

1. kiekvienoje iteracijoje galima iškart naudoti kodą (pvz HTML), leidžiantį atvaizduoti medį;
2. lengvesnis kodo modifikavimas.

Trūkumai:

1. atliekama tiek pat DB prisijungimų kiek ir dirbant rekursijos būdu.

Privalumai paleidžiant steką į SP:

1. kodas vykdomas arčiau duomenų, todėl veikia greičiau;
2. taikomajai programai pateikiamas jau pilnai paruoštas išskleistas medis plokščioje lentelėje, todėl supaprastėja taikomoji programa;
3. tik vienas prisijungimas prie DB.

Trūkumai:

1. ne visose DB/DBMS yra galimybė naudoti SP;
2. sunkiau rašyti kodą, nes jis nutolusioje mašinoje ir rašomas kita kalba.

Kaip pavyzdžiu, Microsoft SQL 6.5 dokumentacijoje pateiktas štai toks kodas, leidžiantis dirbti su hierarchiniais duomenimis ir steku:

```
CREATE PROC expand (@current char(20)) AS
SET nocount on
DECLARE @level int, @line char(20)
CREATE TABLE #stack (item char(20), level int)
INSERT INTO #stack VALUES (@current, 1)
SELECT @level = 1
WHILE @level > 0
BEGIN
  if EXISTS (SELECT * FROM #stack WHERE level = @level)
    BEGIN
      SELECT @current = item
      FROM #stack
      WHERE level = @level
      SELECT @line = space(@level - 1)  @current
      PRINT @line
      DELETE FROM #stack
      WHERE level = @level
        AND item = @current
      INSERT #stack
      SELECT child, @level  1
      FROM hierarchy
```

```
      WHERE parent = @current
      if @@rowcount > 0
        SELECT @level = @level  1
    END
  else
    SELECT @level = @level - 1
END
```

Dabar vėl teoriškai paskaičiuosime šio metodo efektyvumą, kaip tai atlikome rekursiniam metodui.

Pavyzdžiui, sakykime, kad turime tokias sąlygas:

- kiekvieno įrašo gavimas iš DB trunka n sekundžių;
- kiekvienas prisijungimas/atsijungimas prie DB trunka n * 2 sekundžių;
- laikas reikalingas įvykdyti steko operaciją artėja į nulį, lyginant su n;
- saugojamoje procedūroje sukurti laikinąją lentelę trunka n / 2 sekundžių.

Taigi, jei forume turime tuos pačius 1000 pranešimų tai laikai būtų tokie:
naudojant steką taikomoje programoje: (1000 * n) + (1000 * n * 2) = 3000 * n;
naudojant steką išsaugotoje procedūroje: (1000 * n) + (2 * n) + (1000 * (n / 2)) = 1502 * n.

2.2.2 Įdėtosios aibės

Kitas būdas dirbti su medžio struktūromis yra vadinamas "įdėtosios aibės". Keista, bet šis būdas atsirado žymiai vėliau nei pirmasis aprašytas. Šiuo būdu sumodeliuotas medis pranašesnis, nes SQL yra į aibes orientuota kalba [7].

Naudosime tą patį pavyzdinį medį, kurį pateikėme "tėvo-vaiko" modeliu. Aprašant hierarchiją "Įdėtųjų aibių" metodu naudojami tokie laukai: ID, Left, Right.

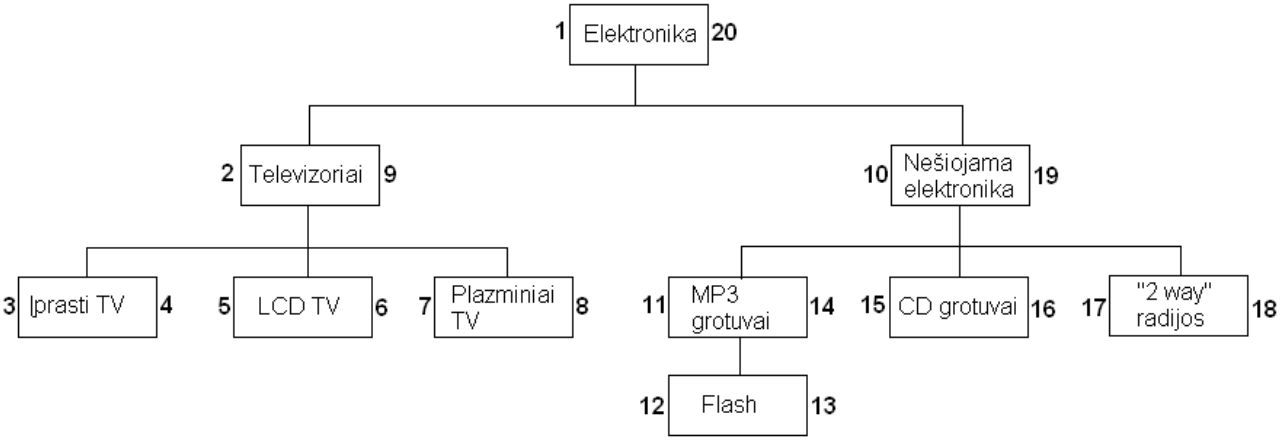
DB lentelės turinys pateiktas 4 lentelėje:

3 Lentelė: DB lentelės turinys - įdėtosios aibės

ID	*Left*	*Right*	*Name*
1	1	20	Elektronika
2	2	9	Televizoriai
3	3	4	Įprastiniai TV
4	5	6	LCD TV
5	7	8	Plazminiai TV
6	10	19	Nešiojama elektronika
7	11	14	MP3 grotuvai

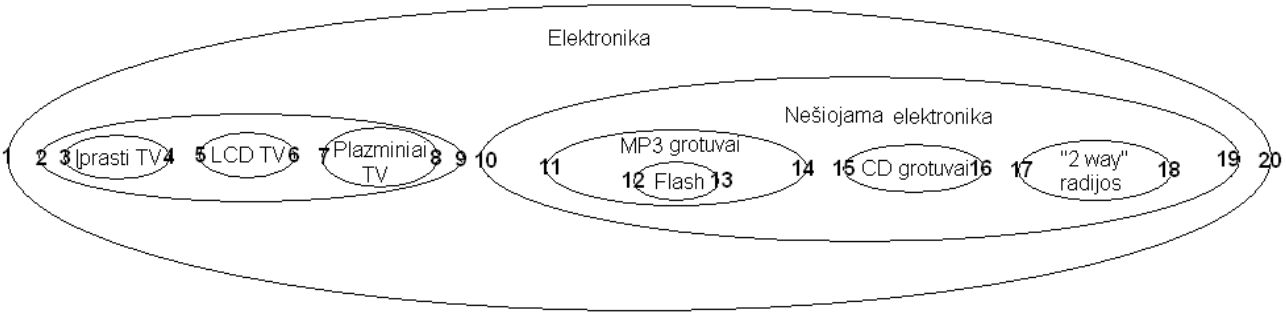
8	12	13	Flash
9	15	16	CD grotuvai
10	17	18	"2 Way" radijo imtuvai

Šitaip sumodeliuotą medį galima grafiškai atvaizduoti bent dviem būdais. 7 paveikslėlyje medžio grafinis vaizdavimas turi net specialų pavadinimą – medžio apibėgimas kontūru (angl. "modified preorder tree traversal algorithm"). Šis būdas taip pavadintas, nes, jei eitume medžio kontūru, matytume, kaip nuosekliai didėja Left/Right numeris.



6 Pav. Įdėtosios aibės - medžio vaizdavimas #1 būdu

Kitas būdas grafiškai atvaizduoti nagrinėjamą medį atspindi pavadinimą "įdėtosios aibės":



7 Pav. Įdėtosios aibės - medžio vaizdavimas #2 būdu

2.2.2.1 Įrašymas

Norint įrašyti naują viršūnę, prieš tai reikia "sukurti vietos", t.y. padidinti numerius, einančius po įterpiamos viršūnės. Tarkime, kad "x" yra naujai kuriamos viršūnės tėvo "Right" numeris. Tada viršūnės įrašymas gali būti realizuotas tokiomis trijomis SQL užklausomis:

```
UPDATE virsunes SET right=right+2 WHERE right>=x;
UPDATE virsunes SET left=left+2 WHERE right>x;
INSERT INTO virsunes (left, right) VALUES(x, x+1);
```

2.2.2.2 Viršūnės trynimas

Viršūnės trynimas yra atvirkščias procesas įrašymui. Vykdomos taip pat trys SQL užklausos (tik atvirkštine tvarka ir atvirkštinio poveikio), pašalinama Viršūnė, o po to panaikinama jos buvusi vieta sumažinant Left/Right numerius. Jei viršūnė turi vaikų tai jie taip pat ištrinami. Sakykime, kad "x" ir "y" yra trinamos viršūnės numeriai atitinkamai "Left" ir "Right". Tada viršūnės trynimas gali būti realizuotas tokiomis trimis SQL užklausomis:

```
DELETE FROM visunes WHERE left=x;
UPDATE virsunes SET right=right-(y-x+1) WHERE right>y;
UPDATE virsunes SET left=left-(y-x+1) WHERE right>y;
```

2.2.2.3 Vaizdavimas

Pagrindinis metodo privalumas yra vaizdavimas. Viena "SELECT... ORDER BY..." užklausa gaunamas visas medis. Taikomojoje programoje medis suformuojamas panaudojant steko struktūrą, panašiai kaip ir atvaizduojant "tėvo-vaiko" modelio medį.

```
setlength(stack, 0);
node := nil;
DataSet := OpenSQLproc('SELECT Left, Right, ID
                        FROM TreeNode
                        ORDER BY  Left ASC');

while not DataSet.Eof do
  begin
  if getlength(stack)>0 then
    begin
    while stack[getlength(stack)-1]<DataSet.FieldByName('Right') do
      begin
      setlength(stack, getlength(stack)-1);

      if node.Parent <> nil then node := node.Parent else node := nil;
      end;
    end;

  PrintNode(DataSet, getlength(stack) );
              // atitraukimo dydis lygus steko dydžiui
              // t.y. getlength(stack)

  setlength(stack, getlength(stack)+1);
  stack[getlength(stack)-1] := DataSet.FieldByName('Right');
  DataSet.Next;
  end;
```

2.2.3 Plokščios lentelės arba kelio saugojimo metodas

Šiuo metodu saugant medžio struktūrą yra išskiriamas atskiras laukas, kuriame saugomas visas kelias nuo šaknės viršūnės iki einamosios viršūnės. Formatuojant kelią panaudojami skyrikliai. Egzistuoja dvi alternatyvos.

Pirmas būdas saugoti kelią yra panaudojant su viršūne susietą informaciją iš dalykinės srities.

4 Lentelė: Kelio saugojimo metodas #1 būdas

ID	*Name*	*Path*
1	Elektronika	/
2	Televizoriai	/Elektronika
3	Įprastiniai TV	/Elektronika/Televizoriai
4	LCD TV	/Elektronika/Televizoriai
5	Plazminiai TV	/Elektronika/Televizoriai
6	Nešiojama elektronika	/Elektronika
7	MP3 grotuvai	/Elektronika/Nešiojama elektronika
8	Flash	/Elektronika/Nešiojama elektronika/MP3 grotuvai
9	CD grotuvai	/Elektronika/Nešiojama elektronika
10	"2 Way" radijo imtuvai	/Elektronika/Nešiojama elektronika

Antras būdas yra viršūnių numeravimas:

5 Lentelė: Kelio saugojimo metodas #1 būdas

ID	*Name*	*Path*
1	Elektronika	1.
2	Televizoriai	1.1.
3	Įprastiniai TV	1.1.1.
4	LCD TV	1.1.2.
5	Plazminiai TV	1.1.3.
6	Nešiojama elektronika	1.2.
7	MP3 grotuvai	1.2.1.
8	Flash	1.2.1.1.
9	CD grotuvai	1.2.2.
10	"2 Way" radijo imtuvai	1.2.3.

Abu būdai (ypač pirmasis) turi tą patį trūkumą – dirbant su didelio gylio medžiu, kelias gali netilpti į jam paskirtą lauką.

2.2.3.1 Įrašymas kai kelias saugo viršūnių pavadinimus

Viršūnės įrašymas yra paprastas, nereikalaujantis duomenų atnaujinimo. Tereikia įvykdyti tik vieną INSERT užklausą:

```
INSERT INTO virsunes (Path, Title) VALUES('kelias', 'pavadinimas');
```

2.2.3.2 Įrašymas kai kelias saugo viršūnių numerius

Viršūnės įrašymas į šiuo būdu sumodeliuotą medį yra ilgas ir sudėtingas procesas, nes reikia atnaujinti viršūnes, einančias po įterpimo vietos. Tam reikalingas reguliariųjų išraiškų (angl. "regular expressions") palaikymas. Viršūnės įrašymo šiuo būdu nenagrinėsime, nes tam nepakanka priemonių, kurias teikia su SQL-92 standartu suderinta DBVS. Tam papildomai reikalingos nestandartinės funkcijos darbui su simbolių eilutėmis, kurios skiriasi priklausomai nuo naudojamos DBVS.

2.2.3.3 Viršūnės trynimas

Viršūnės trynimas iš medžio, kuris kaip kelią saugo viršūnių pavadinimus, yra paprastas procesas, reikalaujantis tik vienos DELETE užklausos. Viršūnės trynimas iš medžio, kuris kaip kelią saugo viršūnių numerius, yra sudėtingas procesas ir nebus nagrinėjamas dėl tų pačių priežasčių kaip ir įrašymas.

2.2.3.4 Viršūnių atnaujinimas

Viršūnių atnaujinimas medyje, kuris kaip kelią saugo viršūnių pavadinimus, reikalauja, kad būtų atnaujinti ir visų vaikų keliai. Šiuo atveju taip pat reikalingos reguliariosios užklausos.

2.2.4 Išvados

Matome, kad nėra vieno uviversalaus, "pačio geriausio" modelio. Kiekvienas modelis turi savų privalumų ir savų trūkumų, todėl kažkurį rinktis reiktų įvertinus reikalavimus sistemai.

- "Kelio saugojimo" modelis geriausiai tinka kai medis vieną kart įrašomas ir daugiau nebekeičiamas. Pritaikymo pavyzdys: saugoti katalogų ir failų hierarchiją.
- "Tėvo-vaiko" modelis geriausiai tinka kai medis turi būti dažnai redaguojamas, t.y. kuriamos ir šalinamos viršūnės.
- "Įdėtųjų aibių" modelis geriausiai tinka kai medis yra labai mažai redaguojamas o reikalingi greiti bei dažni nuskaitymai.

3 Metodų realizavimas pakartotino panaudojimo komponentais, ir jų panaudojimas sistemoje "failų indeksavimas ir paieška"

3.1 Metodų realizavimas – pakartotino panaudojimo komponentų kūrimas

3.1.1 Atkartojimo technologija

3.1.1.1 Programų architektūra

Šiuo metu programos tampa vis didesnės ir sudėtingesnės. Todėl tenka ieškoti efektingesnių būdų sudėtingoms programų struktūroms sukurti. Tam tikslui įgyvendinti kuriamos, vystymo metodikos, taikomas struktūrinis programavimas, vardu sudarymo taisykles, tobulinamas konfigūracijų valdymas ar programinės įrangos architektūra [1]. Programinės įrangos architektūra galima apibrėžti kaip pirminį sistemos planą aukščiausiame abstrakcijos lygyje, apibrėžiantį jos pagrindinius komponentus bei svarbiausius ryšius. Programinės įrangos architektūros leidžia jos kūrėjus atsisakyti smulkesnių architektūros elementų naudojimo, kreipti dėmesį ne į pavienes kodo eilutes, o daugiau dėmesio skirti stambesnių elementų kūrimui bei jų tarpusavio ryšių strūktūrai [3].

3.1.1.2 Objektinis programavimas ir atkartojimo technologija

Objektinis programavimas – tai kitoks programų kūrimo metodas. Pagrindinė priežastis, dėl kurios objektinis programavimas tampa vis populiaresnis, yra auganti atkartojimo technologijos svarba [2]. Naujų sistemų kūrimas ir vystymas yra brangus, o jų priežiūra kainuoja dar daugiau. Tuo tarpu atkartojimo technologija leidžia kurti programinę įrangą pasiremiant jau turimomis programinės įrangos dalimis, komponentais. Tą lengviausia daryti kai programinė įranga yra kuriama būtent atkartojimui, suskaldyta į komponentus.

Atkartojimo technologija teikia tokius privalumus:

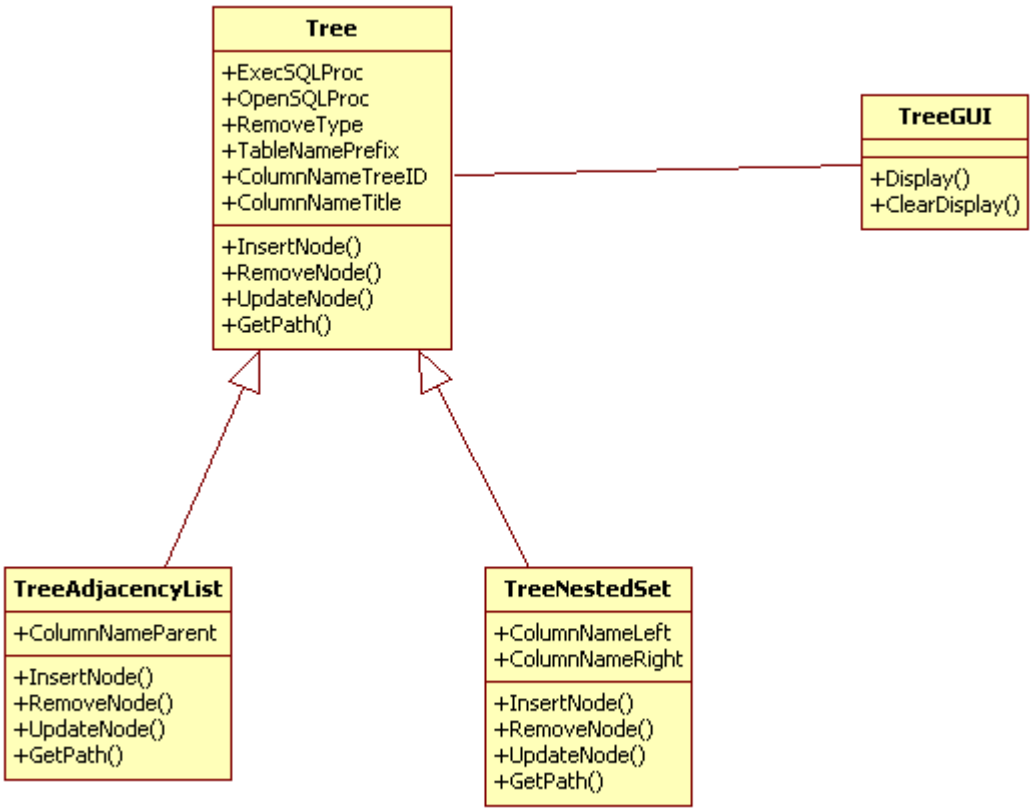
- padidina našumą;
- pakelia kokybę;
- greitesnis kelias į rinką.

3.1.2 Atkartojimo technologija ir tyrinėjamų metodų realizacija

Dėl minėtų atkartojimo technologijos privalumų, metodus, leidžiančius reliacinėse DBVS dirbti su hierarchinėmis struktūromis, realizuosime kaip pakartotino panaudojimo komponentus.

3.2 Detali komponentų architektūra

Klasių diagrama pateikta paveiksėlyje:



8 Pav. Komponentų klasių diagrama

Klasių aprašymas pateiktas A priede.

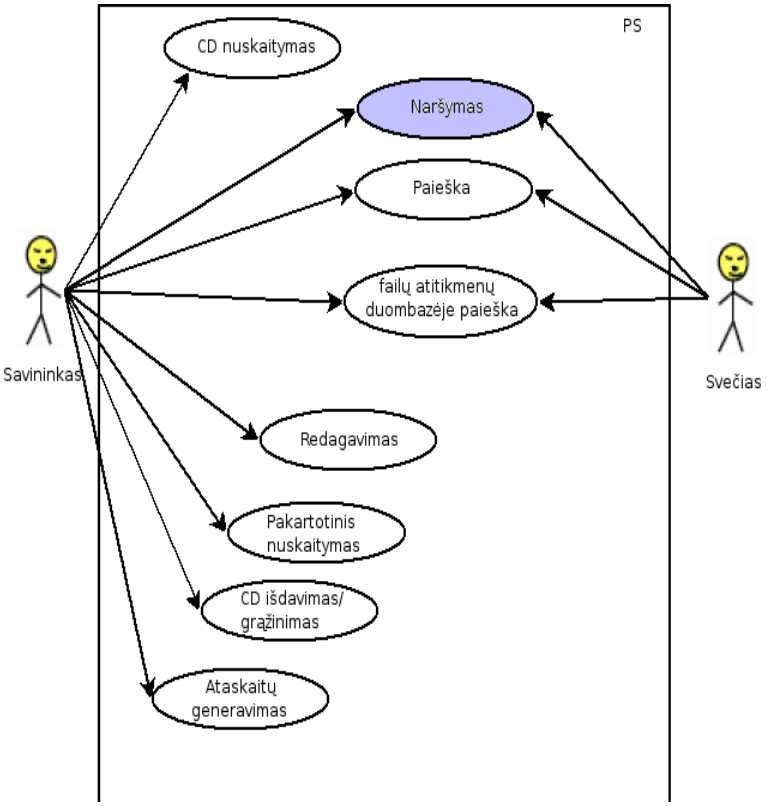
3.3 Failų indeksavimo ir paieškos sistema, naudojanti sukurtus komponentus

3.3.1 Sistemos trumpas apibūdinimas

Sukurta programinė įranga leidžia vesti diskų archyvą. Valdymo balsu funkcija suteikia galimybę naudoti programinę įrangą ir žmonėms su negalią arba tiesiog norintiems padidinti darbo našumą kombinuojant įprastas programinės įrangos valdymo priemones su valdymu balsu.

Programinės įrangos panaudojimo atvejai pateikti 9 paveikslėlyje.

Hierarchinių struktūrų darbo komponentai buvo reikalingi norint realizuoti panaudojimo atvejį „naršymas“.



9 Pav. Panaudojimo atvejų diagrama

3.3.1.1 PS funkcijos

Sistemos vartotojai naudodamiesi programinę įranga galės atlikti tokius veiksmus:

- CD nuskaitymas

leidžia išsaugoti informaciją apie duoto disko turinį (jame esančius failus) į duomenų bazę;

- naršymas

leidžia peržiūrėti į duombazę išsaugoto disko turinį – failų ir katalogų medį, detalesnę informaciją apie failus;

- paieška

leidžia pagal įvestus kriterijus (failo vardas, dydis, tipas ir pnš.) ieškoti failų po diskus, išsaugotus į duombazę;

- koregavimas

leidžia keisti su konkrečiu disku susijusią informaciją;

- pakartotinis CD nuskaitymas

leidžia pakartotinai nuskaityti diską pasikeitus jo turiniui (pvz. perrašius CD-RW diską);

- Disko paskolinimas/grąžinimas

leidžia nurodyti kam diskas paskolinimas (vėliau jis grąžinamas);

- ataskaitų generavimas

leidžia generuoti ataskaitas pagal pasirinktą pjūvį;

- masinė failų atitikmenų duombazėje paieška

leidžia ieškoti ar duombazėje yra failai, esantys nurodytame kataloge.

3.3.1.2 Sistemos kontekstas

Failų indeksavimo ir paieškos posistemė veiks su bet kokia „ANSI SQL-92“ standarto RDBVS. RDBVS galės būti arba lokaliame vartotojo kompiuteryje arba nutolusiame serveryje. Pastaruoju atveju reikalingas pakankamai kokybiškas interneto ryšys, ypač naudojant panaudos atvejus „CD nuskaitymas“ ir „masinė failų atitikmenų duombazėje paieška“.

Kalbos atpažinimo modulis bus sistemos dalis, kuria bus galima pritaikyti ir kituose panašaus tipo projektuose. Šio projekto galutinis produktas bus nepriklausoma kompensuojamosios technikos apskaitos sistema su valdymo balsu funkcija.

3.3.1.3 Vartotojo charakteristikos

Visus sistemos vartotojus suskirstysime į dvi kategorijas (du aktoriai).

- Savininkas – tai kompaktinių diskų archyvo turėtojas.
- Svečias – bet kuris asmuo, kuriam suteiktos duomenų bazės skaitymo teisės.

3.3.1.4 Vartotojo problemos

Vartotojo problemos kurias padės išspręsti kuriamas produktas:

- konkrečiu atveju reikalingų failų radimas kompaktiniuose (ar kt.) diskuose, kai nėra tiesioginio priėjimo prie visų diskų;
- kompaktiniuose diskuose ir kompiuterio kietame diske esančių vienodų failų suradimas;
- kompaktinių diskų skolinimas.

3.3.1.5 Vartotojo tikslai

Programinė įranga turi:

- palengvinti failų po kompaktinius diskus paiešką;
- padėti taupyti vietą, surandant vienodus failus;
- leisti registruoti, kam paskolinamas diskas.

3.3.2 Technologiniai sprendimai

3.3.2.1 Duomenų saugojimas

Duomenų saugojimo metodai.

6 Lentelė: Duomenų saugojimo metodai

Technologija	*Privalumai*	*Trūkumai*
Tekstiniai failai	• nereikia mokytis semantikos	• labai lėtas duomenų apdorojimas
Tipizuoti failai		• sudėtinga realizacija
XML	• paprasta realizacija [14]	• lėtas duomenų apdorojimas
Reliacinės duomenų bazės	• greitas duomenų apdorojimas [6] • laikomąsi standartų • galimybė pasiekti duomenis iš kitos programinės įrangos	

Duomenims saugoti pasirinktos reliacinės duomenų bazės dėl spartaus darbo su duomenimis bei galimybės pasiekti duomenis iš kitos programinės įrangos ir tuo pačiu plėtoti projektą.

Reliacinės duomenų bazės.

7 Lentelė: Reliacinės duomenų bazės

Technologija	*Privalumai*	*Trūkumai*
Konkreti duomenų bazė	• maksimaliai išnaudojamos pasirinktos duomenų bazės funkcijos	• prisirišama prie konkrečios duomenų bazių valdymo sistemos galimybių • mažesnės galimybės integravimui į kitus projektus
Bet kokia duomenų bazė dirbanti „ANSI SQL-92“ standartu [4]	• vartotojas gali pasirinkti tokia duomenų bazę, kokia jam prieinama	• ne maksimaliai išnaudojamos pasirinktos duomenų bazės funkcijos • sudėtingesnis programavimas

Nuspręta kurti produktą kuris veiks su bet kokia „ANSI SQL-92“ standarto RDBVS.

3.3.2.2 Susijungimas su RDBVS

Prisijungimo prie duomenų bazės būdai.

8 Lentelė: Prisijungimo prie RDBVS būdai

DB sąsaja	*Komentarai*
DB teikiama API sąsaja	Šiuo atveju tai neįmanomas pasirinkimas, nes naudojama ne konkreti DB, o bet kokia dirbanti SQL-92 standartu.
Komponentai skirti konkrečiai DB	Šiuo atveju tai neįmanomas pasirinkimas.
BDE sąsaja	Kartu su programine įranga būtina įdiegti BDE biblioteką.
ADO sąsaja	Naudoja ODBC tvarkykles.

Prisijungimui prie duomenų bazės naudojama ADO sąsaja. Ją naudojant, duomenys tarp programos ir duomenų bazės keliauja per ODBC tvarkykles (drivers).

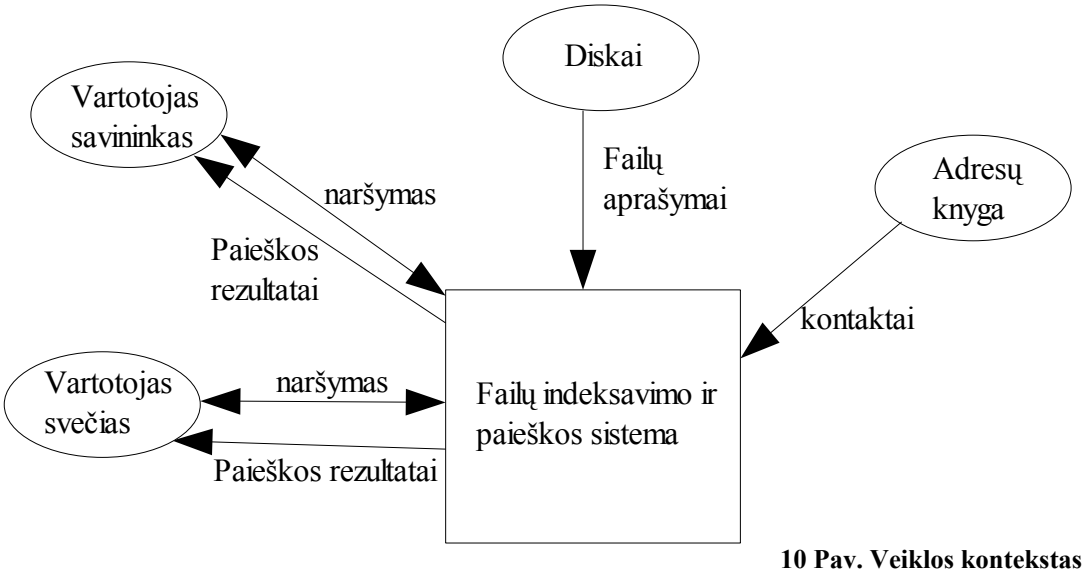
3.3.2.3 Hierarchinės struktūros reliacinėje duomenų bazėje

Darbui su hierarchinėmis struktūromis buvo naudojamas sukurtas pakartotino panaudoji komponentas.

3.3.3 Funkciniai reikalavimai

3.3.3.1 Veiklos kontekstas

Veiklos kontekstas pateiktas 10 paveikslėlyje.



10 Pav. Veiklos kontekstas

3.3.3.2 Veiklos padalinimas

9 Lentelė: Veiklos padalinimas

Eil. Nr.	Įvykio pavadinimas	Įeinantys/išeinantys informacijos srautai
1.	Disko įsigyjimas	Diske esančių failų atributai, aprašymai (in)
2.	Naršymas	Disko, katalogo, failo identifikatorius (in) Failo aprašymas (out)
3.	Paieška per windows programos vartotojo sąsają	Paieškos rezultatai (out)
4.	Paieška per web sąsają	Paieškos rezultatai (out)
5.	Disko skolinimas	Disko ir asmens identifikatorius (in)
6.	Disko grąžinimas	Disko ir asmens identifikatorius (in)

3.3.4 Architektūros specifikacija

Sistemos architektūra pateikiama keliais vaizdais: panaudojimo atvejų (PA), statinis, dinaminis ir išdėstymo. Šie vaizdai yra pateikiami kaip Rational Rose modeliai naudojant unifikuota modeliavimo kalba (UML). Sistemos architektūra pateikta remiantis RUP (Rational Unified Process) rekomendacijomis. Sistemos specifikacija pateikta šiais vaizdais kuriems įgyvendinti reikia UML diagramų:

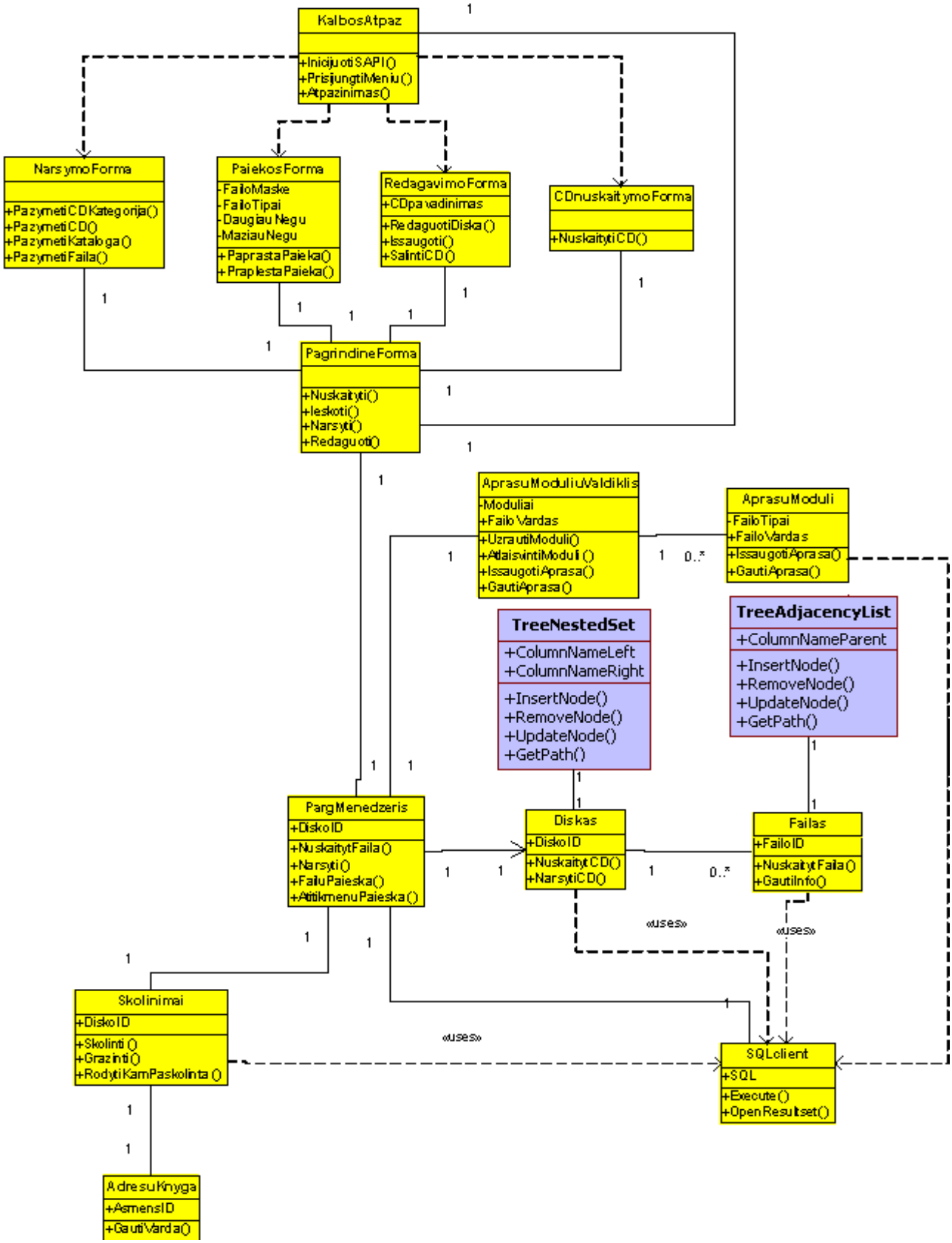
- Panaudojimo atvejų vaizdas (panaudojimo atvejų diagrama).
- Sistemos statinis vaizdas (paketai ir klasių diagramos).
- Sistemos dinaminis vaizdas (būsenų, veiklos, sekų, bendradarbiavimo diagramos).
- Išdėstymo vaizdas (išdėstymo diagrama).

3.3.4.1 Panaudojimo atvejų vaizdas

Pilnas panaudojimo atvejų aprašas pateiktas B priede.

3.3.4.2 Statinis sistemos vaizdas

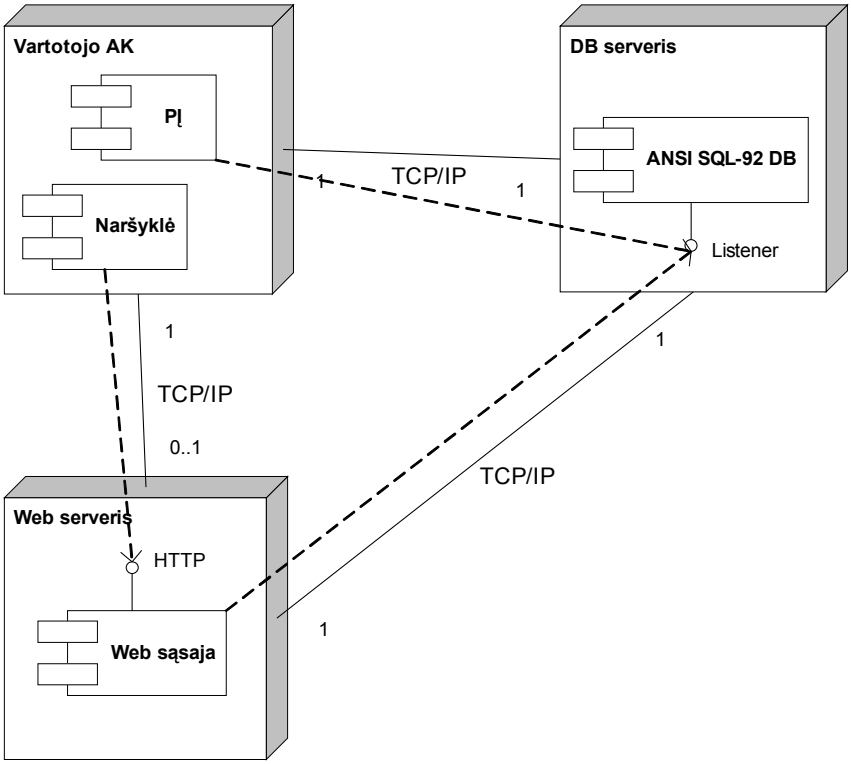
Šis skyrius aprašo loginę sistemos struktūrą. Pateikia sistemos išskaidymą į paketus ir juos sudarančias klases. Komponentai darbui su hierarchinėmis struktūromis pavaizduoti mėlyna spalva.



11 Pav. Bendra klasių diagrama

3.3.4.3 Išdėstymo vaizdas

Paveikslėlyje 17 pateiktas vienas iš išdėstymo variantų. Visgi, tiek web serveris, tiek ir DB serveris gali būti tame pačiame kompiuteryje kaip ir programinė įranga.

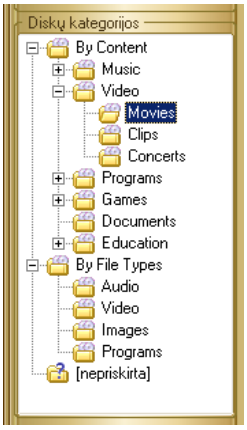


12 Pav. Išdėstymo vaizdas

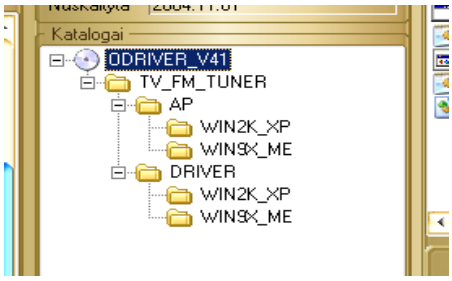
3.3.5 Sukurtų komponentų panaudojimas

Sukurti komponentai buvo panaudoti dviejuose PĮ posistemėse:

- išsaugant disko katalogų ir failų struktūrą (žr. 14 pav.);
- diskų aibę leidžiant sugrupuoti pagal norimus kriterijus, priskiriant juos grupėms, kurios organizuotos medžiu (žr. 13 pav.).



13 Pav. Diskų medis



14 Pav. Failų medis

4 Modelių efektyvumo vertinimas

Palyginsime skirtingų modelių įrašymo ir nuskaitymo greičius. Tam bus prieš ir po kiekvieno eksperimento nuimamas laikas, paskaičiuojamas jų skirtumas. Bus lyginama, kaip šis laikas priklauso nuo:

- medžio lygių skaičiaus;
- nuo santykio, kiek vaikų turi kiekvienas tėvas;
- nuo naudojamos RDBVS.

4.1 Pasiruošimas eksperimentui

Buvo sukurta programa, kuri automatiškai generuoja medžius pagal nurodytus parametrus, ir po to tuos medžius vaizduoja. Parametrai:

- x: koks santykis tėvas:vaikai, t.y. kiek vaikų turi kiekvienas tėvas
- n: kokio lygio medį kurti.

Kad būtų gautas grafikas, eksperimentas kartojamas, paleidžiama iteracija: pirmą kartą kuriamas medis kurio gylis 1, antrą kartą kurio gylis 2 ir t.t. iki n.

Eksperimentų metu bus generuojami du medžių tipai su tokiais parametrais:

1. tėvų:vaikų santykis 1:2, medžio gylis iki 10;
2. tėvų:vaikų santykis 1:5, medžio gylis iki 5.

4.2 Naudojama programinė ir techninė įranga

Visus bandymus darysime po tris kartus ant skirtingos įrangos. Ją suskirstysime į tris profilius.

4.2.1 Profilis #1

Duomenų bazė: lokali, Microsoft Access 2003 su "Microsoft Data Access Components 2.8".

Operacinė sistema: Windows XP Pro SP1.

Techninė įranga:

- procesorius Pentium 4 2,4GHz HT, FSB 800MHz,
- operatyvioji atmintis 1,5GB, 400Mhz, Dual Channel 800MHz.

4.2.2 Profilis #2

Duomenų bazė: lokali, MySQL 4.1.1a su ODBC 3.51.12.

Operacinė sistema: Windows XP Pro SP1.

Techninė įranga:

- procesorius Pentium 4 2,4GHz HT, FSB 800MHz,
- operatyvioji atmintis 1,5GB, 400Mhz, Dual Channel 800MHz.

4.2.3 Profilis #3

Serveris:

Duomenų bazė: nutolusi, MySQL 5.0.22-dev

Operacinė sistema: GNU/Linux Debian 3.0 RC3.

Techninė įranga:

- procesorius Celeron 533MHz, FSB 133MHz,
- operatyvioji atmintis 256MB.

Klientas:

DB sąsaja: ODBC 3.51.12.

Operacinė sistema: Windows XP Pro SP1.

Techninė įranga:

- procesorius Pentium 4 2,4GHz HT, FSB 800MHz,
- operatyvioji atmintis 1,5GB, 400Mhz, Dual Channel 800MHz.

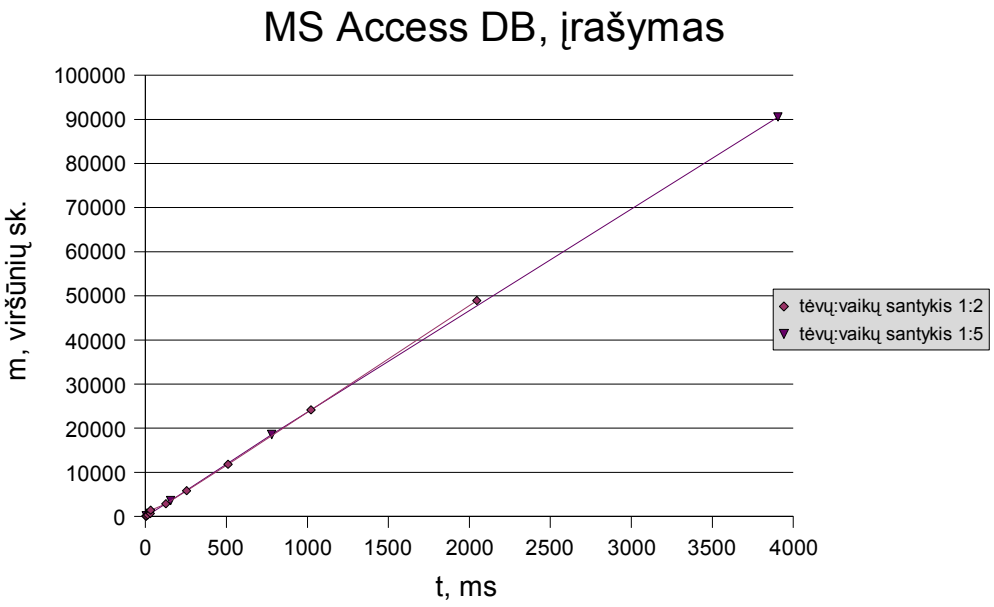
Tinklas:

Dedikuotas 100MB/s Ethernet.

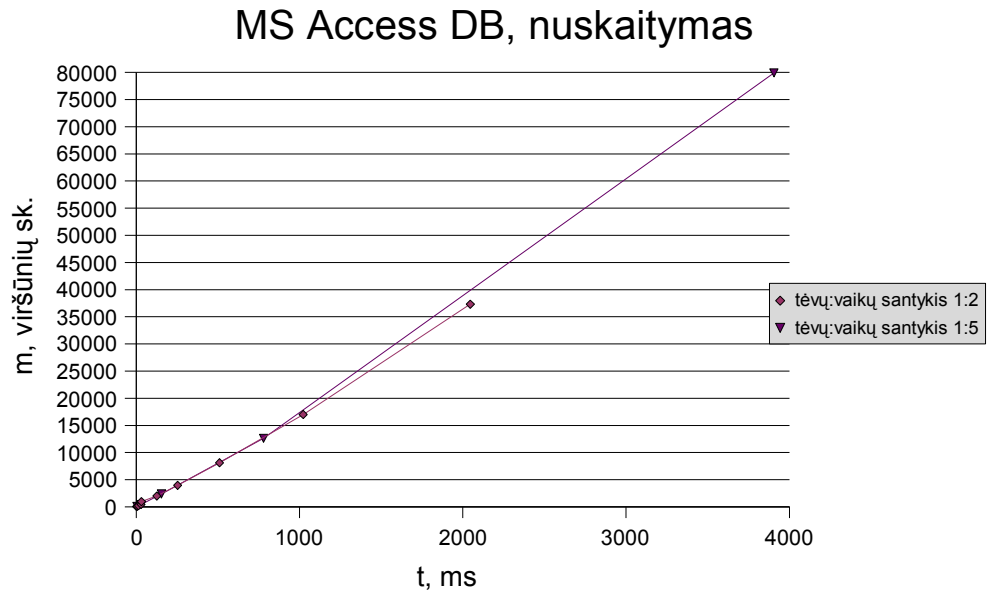
4.3 "tėvo-vaiko" modelio efektyvumo matavimas

Naudojama programinė ir techninė įranga "profilis #1".

Matome 15 ir 16 paveikslėliuose, kad įrašymo laikas tiesiškai priklauso nuo kuriamų viršūnių skaičiaus, ir nepriklauso nuo santykio tėvai:vaikai. Tai galioja ir įrašymui ir nuskaitymui.



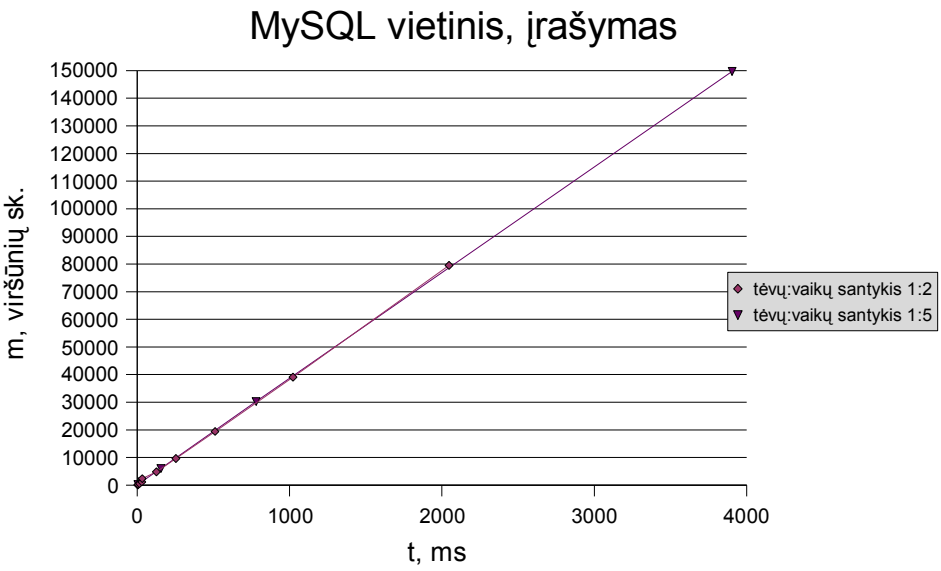
15 Pav. Eksperimentas naudojant lokalią MS Access DB, atliekamas įrašymas



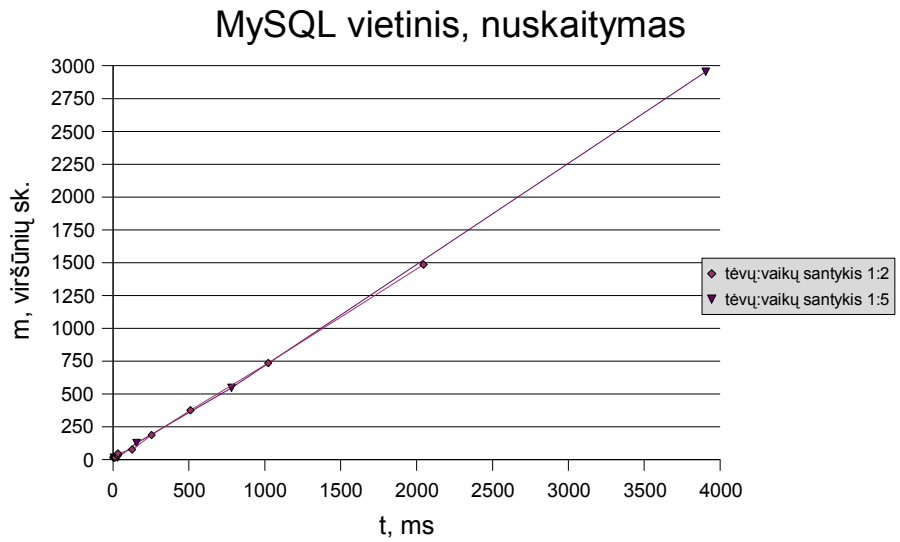
16 Pav. Eksperimentas naudojant lokalią MS Access DB, atliekamas nuskaitymas

Naudojama programinė ir techninė įranga "profilis #2".

Matome 17 ir 18 paveikslėliuose, kad įrašymo laikas tiesiškai priklauso nuo kuriamų viršūnių skaičiaus, ir nepriklauso nuo santykio tėvai:vaikai. Tai galioja ir įrašymui ir nuskaitymui.



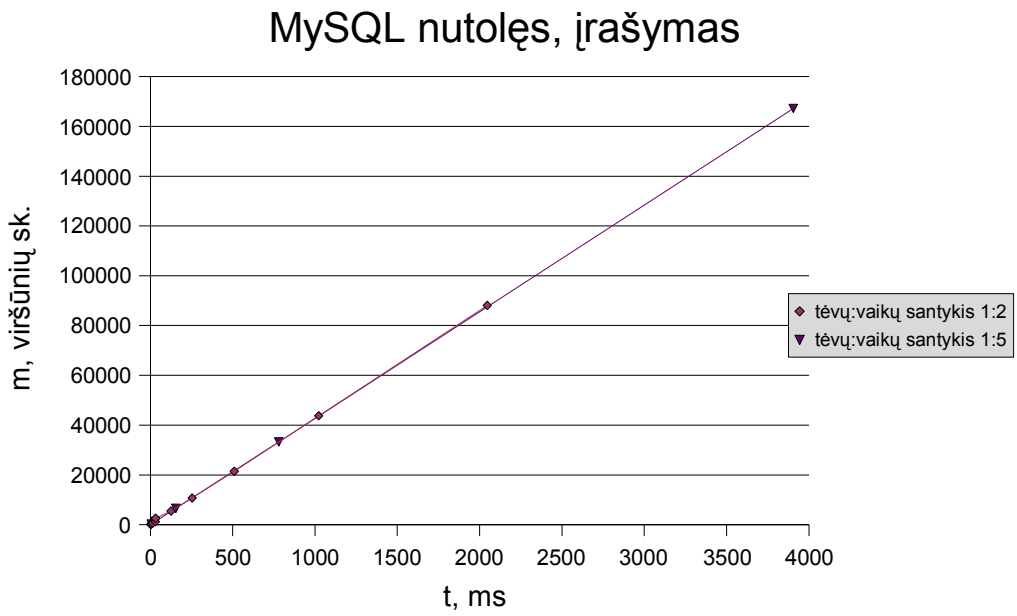
17 Pav. Eksperimentas naudojant lokalią MySQL DB, atliekamas įrašymas



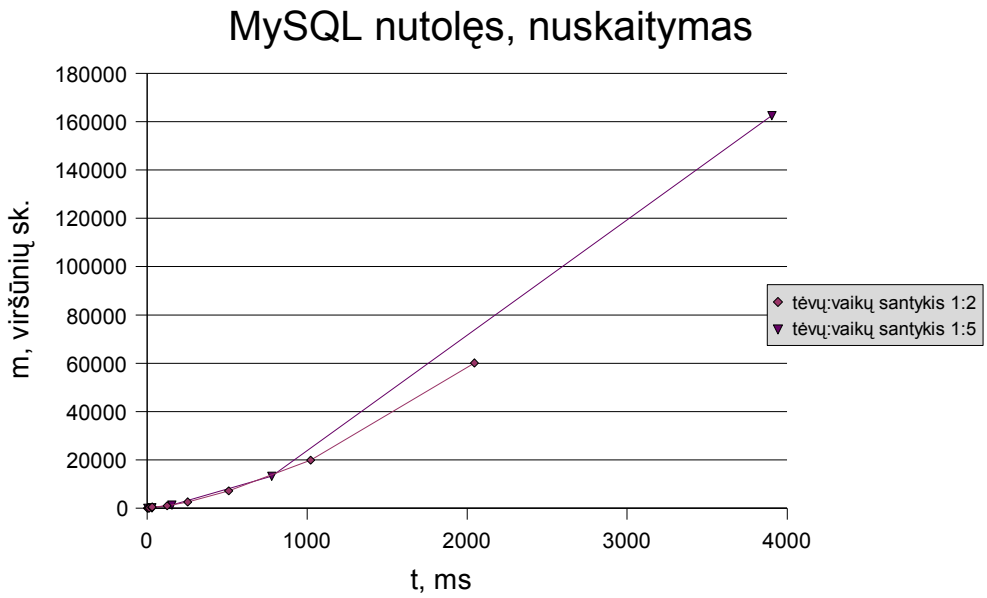
18 Pav. Eksperimentas naudojant lokalią MySQL DB, atliekamas nuskaitymas

Naudojama programinė ir techninė įranga "profilis #3".

Matome 19 ir 20 paveikslėliuose, kad įrašymo laikas tiesiškai priklauso nuo kuriamų viršūnių skaičiaus, ir nepriklauso nuo santykio tėvai:vaikai.



19 Pav. Eksperimentas naudojant nutolusią MySQL DB, atliekamas įrašymas



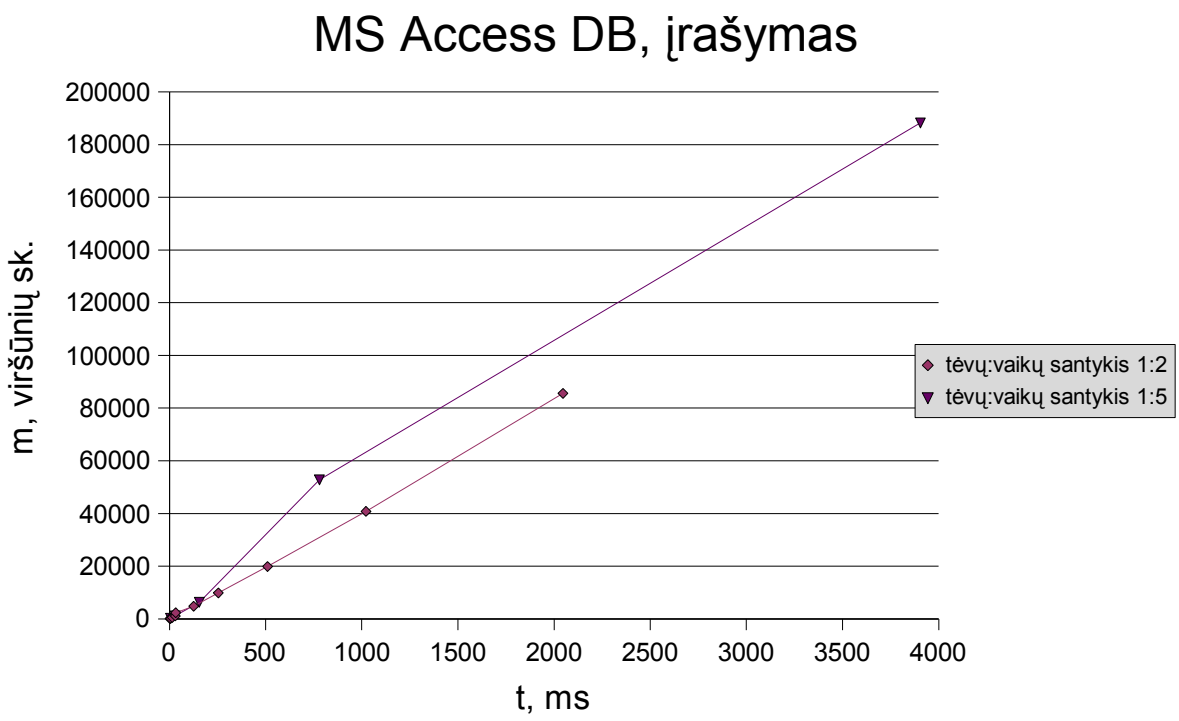
20 Pav. Eksperimentas naudojant nutolusią MySQL DB, atliekamas nuskaitymas

4.4 "Įdėtosios aibės" modelio efektyvumo matavimas

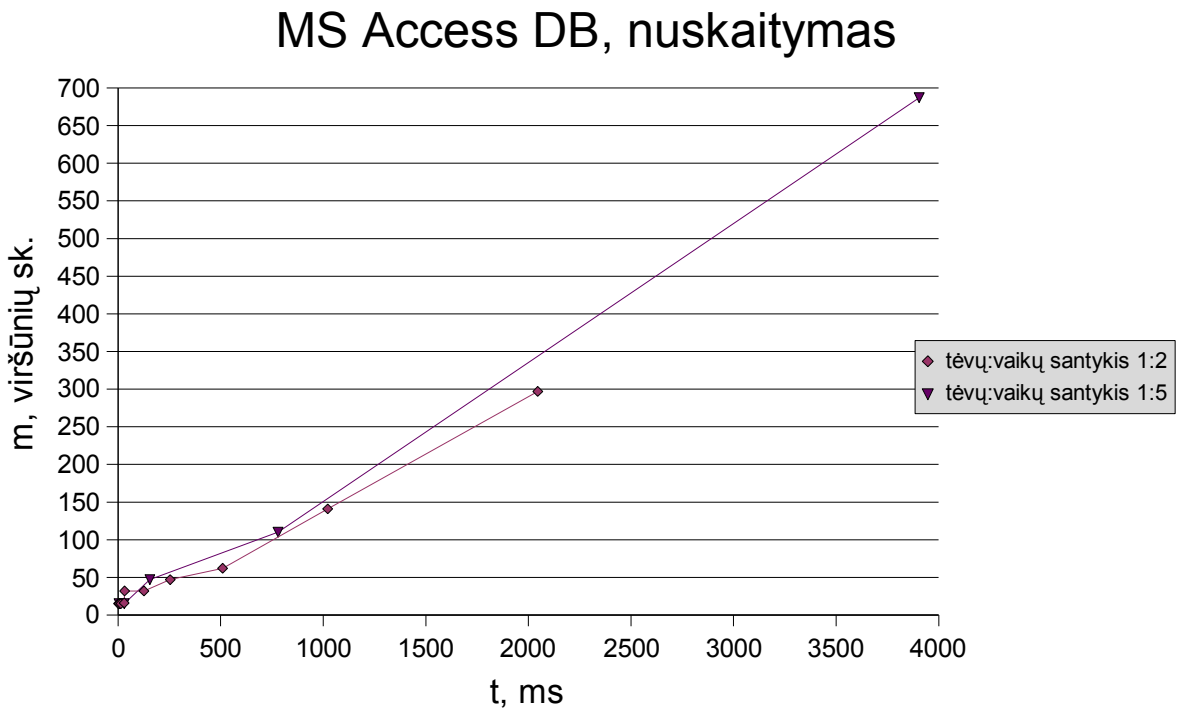
Naudojama programinė ir techninė įranga "profilis #1".

Matome 21 ir 22 paveikslėliuose, kad įrašymo laikas tiesiškai priklauso nuo kuriamų

viršūnių skaičiaus, ir nepriklauso nuo santykio tėvai:vaikai. Tai galioja ir įrašymui ir nuskaitymui.



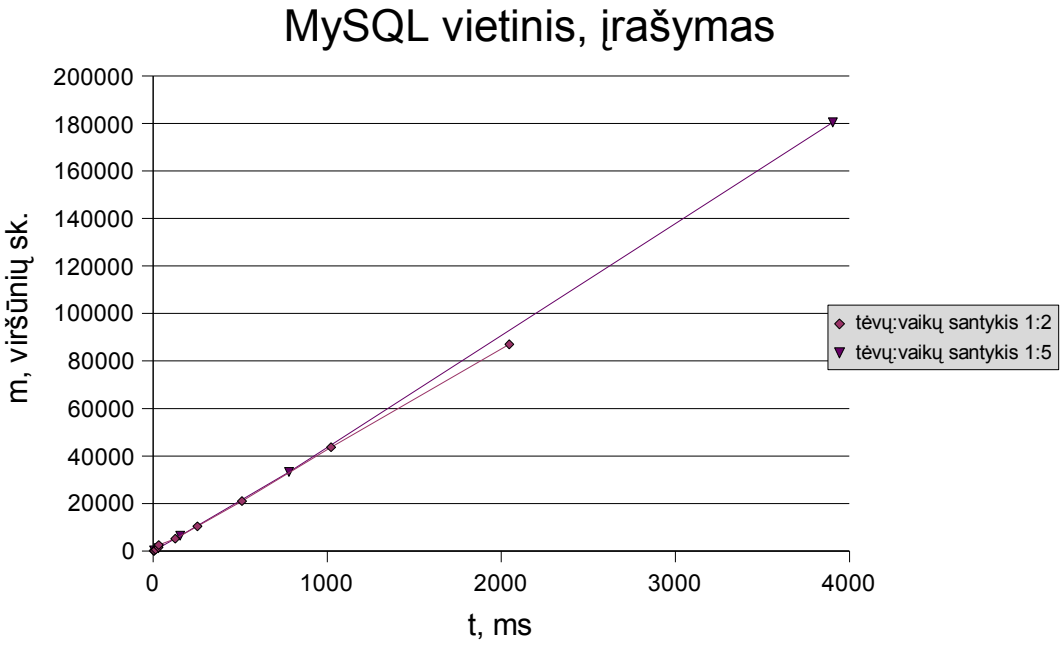
21 Pav. Eksperimentas naudojant lokalią MS Access DB, atliekamas įrašymas



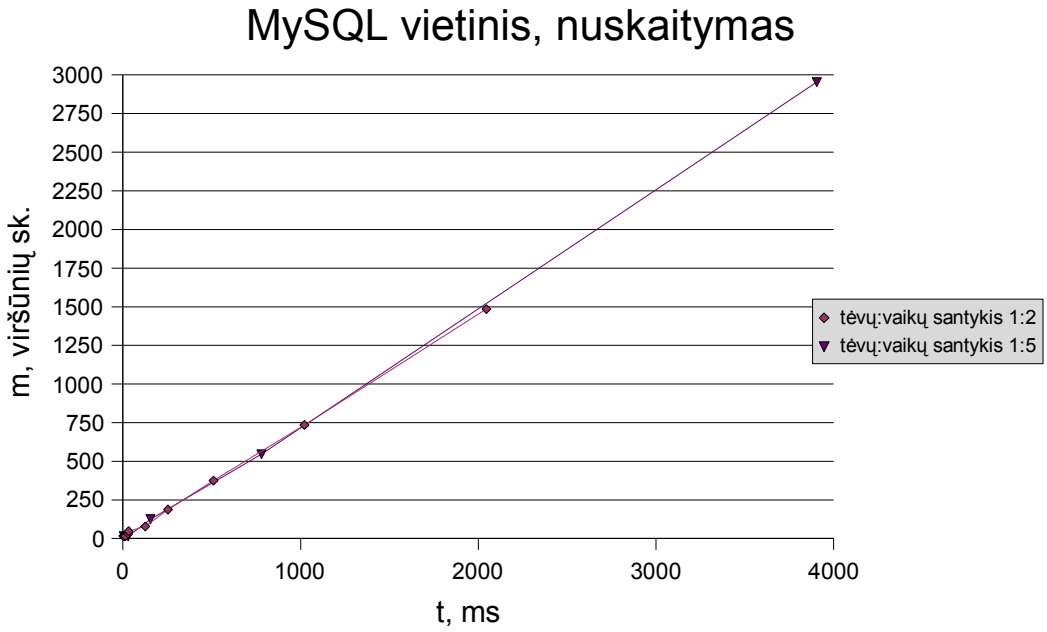
22 Pav. Eksperimentas naudojant lokalią MS Access DB, atliekamas nuskaitymas

Naudojama programinė ir techninė įranga "profilis #2".

Matome 23 ir 24 paveikslėliuose, kad įrašymo laikas tiesiškai priklauso nuo kuriamų viršūnių skaičiaus, ir nepriklauso nuo santykio tėvai:vaikai.



23 Pav. Eksperimentas naudojant lokalią MySQL DB, atliekamas įrašymas

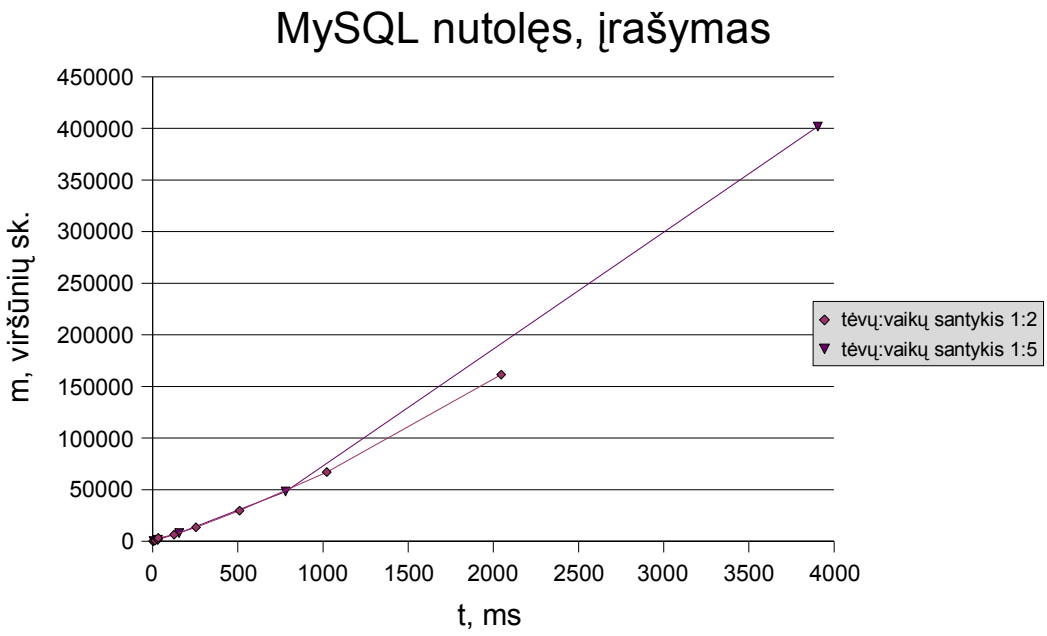


24 Pav. Eksperimentas naudojant lokalią MySQL DB, atliekamas nuskaitymas

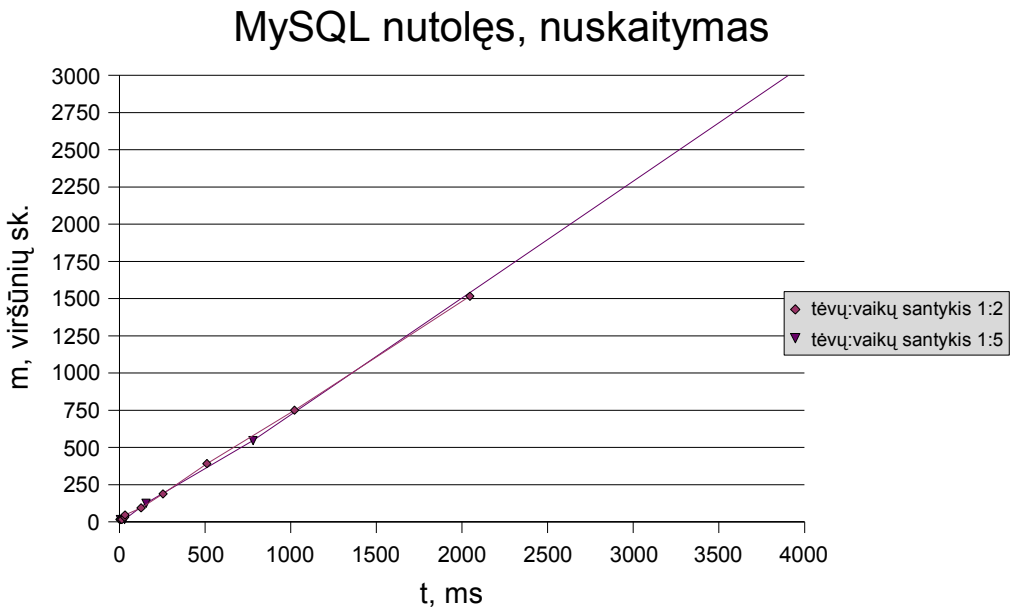
Naudojama programinė ir techninė įranga "profilis #3".

Matome 25 ir 26 paveikslėliuose, kad įrašymo laikas tiesiškai priklauso nuo kuriamų

viršūnių skaičiaus, ir nepriklauso nuo santykio tėvai:vaikai.



25 Pav. Eksperimentas naudojant nutolusią MySQL DB, atliekamas įrašymas



26 Pav. Eksperimentas naudojant nutolusią MySQL DB, atliekamas nuskaitymas

4.5 Eksperimentų išvados

Kaip ir tikėtąsi, paaiškėjo jog tikrai tradicinis "tėvo-vaiko" modelis yra šiek tiek spartesnis kuriant viršūnes, tačiau "įdėtųjų aibių" modelis daug kartų sparčiau nuskaito medį. Šie teiginiai teisingi nepriklausomai nuo to, kokia yra naudojama RDBVS.

"Kelio saugojimo" modelis nebuvo tyrinėjamas, nes jis apjungia pirmųjų dviejų modelių savybes, kas leidžia be tyrimo lengvai išskaičiuoti jo našumą:

- viršūnių įrašymas ir trynimas reikalauja ne tik INSERT bei DELETE užklausų, bet ir UPDATE, kaip ir "įdėtųjų aibių" modelio atveju, bet trunka žymiai ilgiau nes reikalingi veiksmai su reguliariosiomis išraiškomis;
- nuskaitymas toks pat spartus ir nereiklus resursams nes pakanka vienos SELECT užklausos, kaip ir "įdėtųjų aibių" modelio atveju.

5 Išvados

1. Su hierarchinėmis struktūromis mes susiduriame nuolat. Pradedant nuo genealoginio medžio, einant per failų sistemas, ir baigiant forumų antraščių medžiu. Šio darbo turinys yra taip pat medžio struktūra. Deja, jei nenaudojama XML tipo duombazė, tai visi duomenys lentelėse nėra hierarchiniai; tai paprasčiausi plokšti sąrašai.
2. Nėra vieno universalaus modelio darbui su hierarchinėmis struktūromis, kiekvienas turi savų privalumų ir savų trūkumų:
   - "kelio saugojimo" modelis geriausiai tinka kai hierarchija vieną kart įrašoma ir daugiau nebekeičiama. Pritaikymo pavyzdys: saugoti katalogų ir failų hierarchiją;
   - "tėvo-vaiko" modelis geriausiai tinka kai hierarchija turi būti dažnai redaguojama, t.y. kuriamos ir šalinamos viršūnės;
   - "įdėtųjų aibių" modelis geriausiai tinka kai medis yra labai mažai redaguojamas
3. Suprojektuotas pakartotino panaudojimo komponentas, implementuojantis skirtingus darbo su medžiu modelius, buvo panaudotas kuriant programų sistemą „failų indeksavimas ir paieška“. Atkartojimo technologija:
   - padidina PĮ kūrimo proceso našumą;
   - pakelia PĮ kokybę;
   - pagreitina kelią į rinką.
4. Eksperimentai patvirtino, kad:
   - "tėvo-vaiko" modelis atlieka įrašymą 30% sparčiau nei "įdėtųjų aibių" modelis;
   - "įdėtųjų aibių" modelis atlieka medžio pavaizdavimą apie 100-300 kartų greičiau nei "tėvo-vaiko" modelis.

6 Literatūra

[1]: Rikard Land, A Brief Survey of Software Architecture, 2002
[2]: Ralph E. Johnson ir Brian Foote , Designing Reusable Classes, 1988
[3]: Nenad Medvidovic ir David S. Rosenblum, Domains of Concern in Software Architecturesand Architecture Description Languages, 1997
[4]: Romas Baronas, Duomenų bazių sistemos, 2002
[5]: FAT, http://lt.wikipedia.org/wiki/FAT, [žiūrėta 2007-04-01]
[6]: Introduction to Database Systems, http://discovery.bits-pilani.ac.in/discipline/csis/sshekhawat/ISZC332-1.pdf, [žiūrėta 2005-10-14]
[7]: Elsevier Morgan Kaufmann, Joe Celko's Trees and Hierarchies in SQL for Smarties, 2004
[8]: Managing Hierarchical Data in MySQL, http://dev.mysql.com/tech-resources/articles/hierarchical-data.html, [žiūrėta 2007-04-02]
[9]: Michael H. Boyle ir J. Douglas Willms, Multilevel Modelling of Hierarchical Data in Developmental Studies, 2001
[10]: Remembering the father of the relational database, http://www.oracle.com/technology/oramag/oracle/03-jul/o43edit.html, [žiūrėta 2007-04-02]
[11]: System Administrator, Notes for Week 2, Kenneth R. Koehler, http://www.rwc.uc.edu/koehler/admin/week2.html, [žiūrėta 2007-04-01]
[12]: The EXT2 Inode, http://www.science.unitn.it/~fiorella/guidelinux/tlk/node96.html, [žiūrėta 2007-03-25]
[13]: Windows File System Formats, http://book.itzero.com/read/microsoft/0507/microsoft.press.microsoft.windows.internals.fourth.edition.dec.2004.internal.fixed.ebook-ddu_html/0735619174/ch12lev1sec1.html, [žiūrėta 2007-03-18]
[14]: XML vs SQL, RDBMS vs NXD, Internetą: http://www.andrewsavory.com/blog/archives/000139.html, [žiūrėta 2005-11-12]

7 Terminų ir santrumpų žodynėlis

ADO – „ActiveX Data Objects“

BDE – „Borland Database Engine“

DB – duomenų bazė

DBVS – duomenų bazių valdymo sistema

GNU – GNU is Not UNIX

GPL – General Public License

ODBC – laisvo prisijungimo prie duomenų bazės standartas (Open DataBase Connectivity)

OLE – objektų susiejimo standartas (object linking and embedding)

RDBVS – reliacinė duomenų bazių valdymo sistema

RUP – Rational unified process (Rational unifikuotas procesas)

SAPI – kalbos aplikacijų programavimo sąsaja (speech application programming interface)

SQL – Structured Query Language (struktūrizuota užklausų kalba)

UML – Unified Modeling language (unifikuota modeliavimo kalba)

XML – „eXtensible Markup Language“

8 Priedai

Priedas A: detali komponentų architektūra – klasių aprašas.

Tree

Klasifikacija

Klasė

Apibrėžimas

Abstrakti klasė darbui su hierarchine stuktūra duomenų lygyje.

Atsakomybės

Formuoti ir vykdyti SQL užklausas, leidžiančias atlikti veiksmus tokius, kaip viršūnės kūrimas, šalinimas.

Struktūra

Klasės struktūra pateikta klasių diagramoje

Sąveikavimas

Klasės objektas gali funkcionuoti savarankiškai.

Klasę gali naudoti klasė TreeGUI.

Resursai

Naudojama SQL-92 standarto duomenų bazė.

Skaičiavimai

Skaičiavimai pateikti klasės metodų aprašymuose.

Sąsaja/eksportas

Visi *Public* metodai.

Klasės laukai:

ExecSQLProc

Tipas:

Adresas į procedūrą, kurios parametras *string* tipo.

Numatytoji eikšmė:

NULL.

Paskirtis:

Saugo adresą į procedūrą, kuri vykdo SQL užklausą, nurodytą pirmame parametre. Užklausa gali būti tik tokia, kuri negražina reultatų, pvz INSERT, UPDATE.

OpenSQLProc

Tipas:

Adresas į funkciją, kurios parametras *string* tipo, o gražinimo tipas yra DB rezultatų aibė.

Numatytoji eikšmė:

NULL.

Paskirtis:

Saugo adresą į procedūrą, kuri vykdo SQL užklausą, nurodytą pirmame parametre. Užklausa gali būti tik tokia, kuri negražina reultatų aibę, pvz SELECT.

RemoveType

Tipas:

Integer.

Numatytoji eikšmė:

1.

Paskirtis:

Nurodo, kaip reaguoti, jei ištrinama viršūnė turi vaikų:

Reikšmė 1 – išmesti viršūnę su visais vaikais;

Reikšmė 2 – visiems vaikams priskirti naują tėvą – išmetamos viršūnės tėvą;

Reikšmė 3 – pirmąjį vaiką padaryti tėvu.

TableNamePrefix

Tipas:

String.

Numatytoji eikšmė:

''.

Paskirtis:

Nurodo, kad formuojant lentelių pavadinimus, reikia papildomai pradžioje pridėti TableNamePrefix.

ColumnNameTreeID

Tipas:

String.

Numatytoji eikšmė:

'ctreeid'.

Paskirtis:

Nurodo pavadinimą lauko, saugančio medžio, su kuriuo dirbama, ID reikšmę.

ColumnNameTitle

Tipas:

String.

Numatytoji eikšmė:

'ctitle'.

Paskirtis:

Nurodo pavadinimą lauko, kuriame tekstiniu formatu saugomas viršūnės pavadinimas.

Klasės metodai:

InsertNode();

Apibrėžimas:

Abstraktus metodas viršūnės sukūrimui.

Atsakomybės:

Nurodytai viršūnei sukurti vaiką. Jei viršūnė nenurodyta, tada šakninei viršūnei sukurti vaiką.

Sąveikavimas:

Metodas bendros paskirties (Public).

RemoveNode();

Apibrėžimas:

Abstraktus metodas viršūnės pašalinimui.

Atsakomybės:

Pašalinti nurodytą viršūnę.

Sąveikavimas:

Metodas bendros paskirties (Public).

UpdateNode();

Apibrėžimas:

Viršūnės pervadinimui.

Atsakomybės:

Pervadinti viršūnę.

Sąveikavimas:

Metodas bendros paskirties (Public).

GetPath();

Apibrėžimas:

Abstraktus metodas kelio iki viršūnės gražinimui *string* formatu.

Atsakomybės:

Rasti ir suformuoti kelią iki viršūnės.

Sąveikavimas:

Metodas bendros paskirties (Public).

TreeAdjacencyList

Klasifikacija

Klasė

Apibrėžimas

Klasė (išvesta iš „Tree“ klasės) darbui su hierarchine stuktūra duomenų lygyje, realizuojanti

„tėvo-vaiko“ modelį.

Atsakomybės

Formuoti ir vykdyti SQL užklausas, leidžiančias atlikti veiksmus tokius, kaip viršūnės kūrimas, šalinimas.

Struktūra

Klasės struktūra pateikta klasių diagramoje

Sąveikavimas

Klasės objektas gali funkcionuoti savarankiškai.

Klasę gali naudoti klasė TreeGUI.

Resursai

Naudojama SQL-92 standarto duomenų bazė.

Skaičiavimai

Skaičiavimai pateikti klasės metodų aprašymuose.

Sąsaja/eksportas

Visi *Public* metodai.

Klasės laukai:

ColumnNameParent

Tipas:

String.

Numatytoji eikšmė:

'cparent'.

Paskirtis:

Nurodo pavadinimą lauko, kuris saugo elemento tėvą.

Klasės metodai:

InsertNode();

Apibrėžimas:

Abstraktaus metodo implementacija viršūnės sukūrimui.

Atsakomybės:

Nurodytai viršūnei sukurti vaiką. Jei viršūnė nenurodytas, tada šakninei viršūnei sukurti vaiką.

Sąveikavimas:

Metodas bendros paskirties (Public).

RemoveNode();

Apibrėžimas:

Abstraktaus metodo implementacija viršūnės pašalinimui.

Atsakomybės:

Pašalinti nurodytą viršnę.

Sąveikavimas:

Metodas bendros paskirties (Public).

UpdateNode();

Apibrėžimas:

Abstraktaus metodo implementacija viršūnės pervadinimui.

Atsakomybės:

Pervadinti viršūnę.

Sąveikavimas:

Metodas bendros paskirties (Public).

GetPath();

Apibrėžimas:

Abstraktaus metodo implementacija kelio iki viršūnės gražinimui *string* formatu.

Atsakomybės:

Rasti ir suformuoti kelią iki viršūnės.

Sąveikavimas:

Metodas bendros paskirties (Public).

TreeNestedSet

Klasifikacija

Klasė

Apibrėžimas

Klasė (išvesta iš „Tree“ klasės) darbui su hierarchine stuktūra duomenų lygyje, realizuojanti „įdėtųjų aibių“ modelį.

Atsakomybės

Formuoti ir vykdyti SQL užklausas, leidžiančias atlikti veiksmus tokius, kaip viršūnės kūrimas, šalinimas.

Struktūra

Klasės struktūra pateikta klasių diagramoje

Sąveikavimas

Klasės objektas gali funkcionuoti savarankiškai.

Klasę gali naudoti klasė TreeGUI.

Resursai

Naudojama SQL-92 standarto duomenų bazė.

Skaičiavimai

Skaičiavimai pateikti klasės metodų aprašymuose.

Sąsaja/eksportas

Visi *Public* metodai.

Klasės laukai:

ColumnNameLeft

Tipas:

String.

Numatytoji eikšmė:

'cleft'.

Paskirtis:

Nurodo pavadinimą lauko, saugančio „left“ reikšmę.

ColumnNameRight

Tipas:

String.

Numatytoji eikšmė:

'cright'.

Paskirtis:

Nurodo pavadinimą lauko, saugančio „right“ reikšmę.

Klasės metodai:

InsertNode();

Apibrėžimas:

Abstraktaus metodo implementacija viršūnės sukūrimui.

Atsakomybės:

Nurodytai viršūnei sukurti vaiką. Jei viršūnė nenurodyta, tada šakninei viršūnei sukurti vaiką.

Sąveikavimas:

Metodas bendros paskirties (Public).

RemoveNode();

Apibrėžimas:

Abstraktaus metodo implementacija viršūnės pašalinimui.

Atsakomybės:

Pašalinti nurodytą viršūnę.

Sąveikavimas:

Metodas bendros paskirties (Public).

UpdateNode();

Apibrėžimas:

Abstraktus metodo implementacija viršūnės pervadinimui.

Atsakomybės:

Pervadinti viršūnę.

Sąveikavimas:

Metodas bendros paskirties (Public).

GetPath();

Apibrėžimas:

Abstraktus metodo implementacija kelio iki viršūnes gražinimui *string* formatu.

Atsakomybės:

Rasti ir suformuoti kelią iki viršūnės.

Sąveikavimas:

Metodas bendros paskirties (Public).

TreeGUI

Klasifikacija

Klasė

Apibrėžimas

Klasė darbui su grafine vartotojo sąsaja.

Atsakomybės

Sinchronizuoti duomenis tarp „Tree“ klasės objekto ir „TTreeView“ komponento.

Struktūra

Klasės struktūra pateikta klasių diagramoje

Sąveikavimas

Klasės objektas naudoja objektą paveldėtą iš „Tree“ klasės.

Resursai

Naudojama SQL-92 standarto duomenų bazė.

Skaičiavimai

Skaičiavimai pateikti klasės metodų aprašymuose.

Sąsaja/eksportas

Visi „Public“ metodai.

Klasės metodai:

Display();

Apibrėžimas:

Metodas, atvaizduojantis medžio struktūrą „TTreeView“ objekte.

Atsakomybės:

Atvaizduojanti medžio struktūrą „TTreeView“ objekte.

Sąveikavimas:

Metodas bendros paskirties (Public).

ClearDisplay();

Apibrėžimas:

Metodas, išvalantis medžio struktūrą „TTreeView“ objekte. Metodas automatiškai iškviečiamas klasės destruktoriuje.

Atsakomybės:

Pašalinti iš „TTreeView“ objekto visas viršūnes ir atlaisvinti jiems išskirtą papildomų struktūrų atmintį.

Sąveikavimas:

Metodas bendros paskirties (Public).

Priedas B: panaudojimo atvejų aprašymai.

10 Lentelė: Panaudojimo atvejis #1 – CD nuskaitymas

Panaudojimo atvejis	CD nuskaitymas
Tikslas	Išsaugoti informaciją apie duoto disko turinį (jame esančius failus) į duomenų bazę.
Aktoriai	Savininkas
Ryšiai su kitais panaudojimo atvejais	Redagavimas
Prieš sąlygos	Nėra
Sužadinimo salyga	Aktorius paleidžia diskų nuskaitymo vedlį.
Po sąlygos	Duombazė papildyta duoto disko failų atributais.
Pagrindinis scenarijus	1. Įdedamas diskas į CD/DVD nuskaitymo įrenginį. 2. Paleidžiamas diskų nuskaitymo vedlys. 3. Atliekami nurodyti žingsniai.
Alternatyvūs scenarijai	• Nepasibaigus disko nuskaitymui, vartotojas išjungia vedlį. Tuo atveju duombazės turinys privalo būti nepakitęs lyginant būsenas prieš ir po veiksmo.

11 Lentelė: Panaudojimo atvejis #2 – naršymas

Panaudojimo atvejis	Naršymas
Tikslas	Peržiūrėti į duombazę išsaugoto disko turinį – failų ir katalogų medį, detalesnę informaciją apie failus.
Aktoriai	Savininkas Svečias
Ryšiai su kitais panaudojimo atvejais	Nėra
Prieš sąlygos	Duomenų bazėje išsaugotas bent vienas diskas
Sužadinimo salyga	Aktorius atveria naršymo langą.
Po sąlygos	Duombazės informacija lieka nepakitusi.
Pagrindinis scenarijus	1. Atveria naršymo langas. 2. Vaikštoma per failų-katalogų medį.
Alternatyvūs scenarijai	Nėra

12 Lentelė: Panaudojimo atvejis #3 – paieška

Panaudojimo atvejis	Paieška
Tikslas	Pagal įvestus kriterijus (failo vardas, dydis, tipas ir pnš.) ieškoti failų po diskus, išsaugotus į duombazę.
Aktoriai	Savininkas Svečias
Ryšiai su kitais panaudojimo atvejais	Nėra
Prieš sąlygos	Duomenų bazėje išsaugotas bent vienas diskas
Sužadinimo salyga	Aktorius atveria paieškos langą.
Po sąlygos	Duombazės informacija lieka nepakitusi.

Panaudojimo atvejis	Paieška
Pagrindinis scenarijus	1. Atveriamas paieškos langas. 2. Įvedami paieškos kriterijai. 3. Vykdoma paieška.
Alternatyvūs scenarijai	Aktorius vykdo paiešką neįvedęs paieškos parametrų – parodomas pranešimas juos įvesti.

13 Lentelė: Panaudojimo atvejis #4 – failų atitikmenų paieška

Panaudojimo atvejis	Masinė failų atitikmenų duombazėje paieška
Tikslas	Ieškoti ar duombazėje yra nurodytame kataloge esantys failai.
Aktoriai	Savininkas Svečias
Ryšiai su kitais panaudojimo atvejais	Nėra
Prieš sąlygos	Duomenų bazėje išsaugotas bent vienas diskas.
Sužadinimo salyga	Aktorius aktyvuoja atitikmenų paiešką.
Po sąlygos	Nėra
Pagrindinis scenarijus	1. Nuromas katalogas, kuriame esantiems failams ieškoti atitikmenų duombazėje. 2. Vykdoma atitikmenų paieška.
Alternatyvūs scenarijai	Nėra

14 Lentelė: Panaudojimo atvejis #5 – koregavimas

Panaudojimo atvejis	Koregavimas
Tikslas	Keisti su konkrečiu disku susijusią informaciją.
Aktoriai	Savininkas
Ryšiai su kitais panaudojimo atvejais	Nėra
Prieš sąlygos	Duomenų bazėje išsaugotas bent vienas diskas
Sužadinimo salyga	Aktorius atveria redagavimo langą.
Po sąlygos	Pakeista informacija apie diską.
Pagrindinis scenarijus	1. Atveriamas redagavimo langas. 2. Pasirenkamas diskas. 3. Spaudžiamas mygtukas „redaguoti“. 4. Koreguojama informacija apie diską. 5. Išsaugomi pakeitimai nuspaudus mygtuką „Įrašyti“.
Alternatyvūs scenarijai	Pasirinktas diskas pašalinamas iš duomenų bazės.

15 Lentelė: Panaudojimo atvejis #6 – pakartotinis nuskaitymas

Panaudojimo atvejis	Pakartotinis nuskaitymas
Tikslas	Pakartotinai nuskaityti diską pasikeitus jo turiniui (pvz. perrašius CD-RW diską).
Aktoriai	Savininkas
Ryšiai su kitais panaudojimo atvejais	Redagavimas
Prieš sąlygos	Duomenų bazėje išsaugotas bent vienas diskas

Sužadinimo salyga	Aktorius aktyvuoja pakartotinį disko nuskaitymą.
Po sąlygos	Pakeista informacija apie disko turinį.
Pagrindinis scenarijus	1. Atveriamas redagavimo langas. 2. Pasirenkamas diskas. 3. Spaudžiamas mygtukas „skaityti iš naujo“.
Alternatyvūs scenarijai	Nėra

16 Lentelė: Panaudojimo atvejis #7 – disko paskolinimas (išdavimas) / grąžinimas

Panaudojimo atvejis	Disko paskolinimas/grąžinimas
Tikslas	Nurodyti kam diskas paskolinimas. Vėliau jis grąžinamas.
Aktoriai	Savininkas
Ryšiai su kitais panaudojimo atvejais	Nėra
Prieš sąlygos	Duomenų bazėje išsaugotas bent vienas diskas.
Sužadinimo salyga	Aktorius nuspaudžia mygtuką „skolinti“/„grąžinti“.
Po sąlygos	Disko skolinimo žurnalas papildomas nauju įrašu.
Pagrindinis scenarijus	1. Atveriamas redagavimo langas. 2. Pasirenkamas diskas. 3. Spaudžiamas mygtukas „skolinti/grąžinti“.
Alternatyvūs scenarijai	Nėra