KAUNO TECHNOLOGIJOS UNIVERSITETAS

FUNDAMENTALIŲJŲ MOKSLŲ FAKULTETAS

TAIKOMOSIOS MATEMATIKOS KATEDRA

Skaistė Baršauskaitė

LIETUVOS AKCIJŲ RINKOS PASIŪLOS IR PAKLAUSOS SRAUTŲ ANALIZĖ

Magistro darbas

Vadovas
doc. dr. G. Račkauskas

KAUNAS, 2008

KAUNO TECHNOLOGIJOS UNIVERSITETAS

FUNDAMENTALIŲJŲ MOKSLŲ FAKULTETAS

TAIKOMOSIOS MATEMATIKOS KATEDRA

TVIRTINU
Katedros vedėjas
doc. dr. N. Listopadskis

2008 06 06

LIETUVOS AKCIJŲ RINKOS PASIŪLOS IR PAKLAUSOS SRAUTŲ ANALIZĖ

Taikomosios matematikos magistro baigiamasis darbas

Vadovas
doc. dr. G. Račkauskas
2008 05 30

Recenzentė
Laurita Petrošienė
2008 06 02

Atliko
FMMM - 6 gr. stud.
S. Baršauskaitė
2008 05 23

KAUNAS, 2008

KVALIFIKACINĖ KOMISIJA

Pirmininkas: Leonas Saulis, profesorius (VGTU)

Sekretorius: Eimutis Valakevičius, docentas (KTU)

Nariai: Algimantas Jonas Aksomaitis, profesorius (KTU)

Vytautas Janilionis, docentas (KTU)

Vidmantas Povilas Pekarskas, profesorius (KTU)

Rimantas Rudzkis, habil. dr., valdybos pirmininko pavaduotojas (DnB NORD Bankas)

Zenonas Navickas, profesorius (KTU)

Arūnas Barauskas, dr., vice-prezidentas projektams (UAB „Baltic Amadeus“)

Baršauskaitė S. The analysis of quoted bid-ask spread of Vilnius Stock Exchange: Master's work in applied mathematics / supervisor dr. assoc. prof. G. Račkauskas; Department of Applied mathematics, Faculty of Fundamental Sciences, Kaunas University of Technology. – Kaunas, 2008. – 100 p.

SUMMARY

Nine types of stock were chosen to analyse quoted bid-ask spread of Vilnius Stock Exchange. According to the value and number of transactions of the stock, it can be divided into three groups: non-liquid, half-liquid and liquid stock. Public market depth information and data of trade was taken from Vilnius Stock Exchange website http://www.baltic.omxgroup.com/ during the period from 25th February 2008 to 18th April 2008.

In my work I have analysed inside bid-ask spread, effective spread and fixed prices. Roll measure was measured using trade prices of stock; stationarity of differences of trade prices were examined using reverse arrangement test. As had been expected, I came to conclusion that the stock market for chosen stocks is informationally inefficient. Due to this reason the Roll measure is not correct.

By using C++ programming language the following programming tools were created:

- Data reading from internet tool;
- Data collection and correction tool;
- Data analysis tool.

SANTRAUKA

Lietuvos akcijų rinkos pasiūlos ir paklausos srautų analizei buvo pasirinktos 9 akcijos. Jas, pagal įvykusių sandorių skaičių ir vertę, galima suskirstyti į tris grupes: nelikvidžios, pusiau likvidžios ir likvidžios akcijos. Šių akcijų, viešai skelbiami rinkos gylio ir įvykusių sandorių, duomenys buvo imami iš Vilniaus vertybinių popierių biržos internetinio puslapio http://www.baltic.omxgroup.com/ nuo 2008 02 25 iki 2008 04 18.

Darbe buvo skaičiuojami paprastas (*inside bid-ask spread*), efektyvus (*effective spread*) ir užfiksuotas kainų skirtumai. Roll matas skaičiuojamas remiantis akcijų įvykusių sandorių kainomis, kurių skirtumų stacionarumas ištirtas RA-kriterijumi (*reverse arrangement test*). Kaip ir buvo galima tikėtis, parodyta, kad pasirinktų akcijų rinka yra neefektyvi. Dėl šios priežasties, kiekvienai akcijai apskaičiuotas Roll matas yra labai grubus.

Naudojant C++ programavimo kalbą, buvo sukurta programinė įranga:

- Duomenų skaitymui iš interneto;
- Duomenų bazės kūrimui, apdorojimui ir redagavimui;
- Duomenų analizei.

Gauti rezultatai pristatyti VII-oje studentų taikomosios matematikos konferencijoje. Taip pat paruoštas pranešimas Lietuvos matematikų draugijos XLIX konferencijai.

TURINYS

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

ĮVADAS

Akcijų rinkoje pasiūlos ir paklausos srautai atspindi investuotojų lūkesčius, akcijos kainos kitimo tendencijas. Pagrindinis bet kurios finansų rinkos analizės tikslas yra aptikti tinkamus investavimui instrumentus, bandant nuspėti, ar jų vertė didės, ar mažės, ir kada tai įvyks. Techninė analizė apibūdinama kaip paskelbtų (istorinių) rinkos duomenų naudojimas tam tikro finansinio instrumento rinkos analizei ir prognozei. Ši analizė teigia, kad svarbesni yra trumpalaikiai ir psichologiniai faktoriai. Visiškai priešingai techninei analizei yra efektyvios rinkos hipotezė, kuri teigia, kad finansinių instrumentų kainos jau atspindi jų tikrąją, ekonominę vertę, kuri yra pagrįsta faktine informacija. Jeigu finansų rinkos veikia efektyviai ir yra gerai informuotos, tai rinkos kaina nedaug tenukrypsta nuo esminės, tikrosios, ekonominės vertės. Fundamentalioji analize, kuri siekia rasti finansinio instrumento tikrąja vertę ir palyginti ją su rinkos verte, kad būtų galima sužinoti, kokį sprendimą reikia daryti, sako, kad investuotojai iš tų pačių duomenų gali daryti skirtingas išvadas ar įžvalgas ir jei vieno jos geresnės nei kitu, tai jis gali gauti didesnį pelną. Tai, kad kai kurie investuotojai netiki efektyvios rinkos hipoteze, padeda rinkai būti efektyvia. Ieškodami nepakankamai įvertintų arba pervertintų akcijų, investuotojai suranda informaciją ir veikia kuo greičiau. Beieškodami informacijos ir besistengdami aplenkti kitus, jie daro rinka efektyvesne. Kasdien atliekama fundamentalioji analizė ir atsispindi kainose.

Darbe plačiau nagrinėjama efektyvios rinkos hipotezė. Kaip vienas iš rinkos efektyvumo kriterijų buvo naudojamas Roll[1] matas [5]. Jo taikymas užsienio akcijų rinkoje (NASDAQ) ir gauti rezultatai pateikiami [6] straipsnyje.

Pagrindiniai darbo tikslai:

- Vilniaus vertybinių popierių biržos viešai skelbiamų duomenų panaudojimo galimybių išplėtimas;
- Teorinių samprotavimų apie rinkos efektyvumą pritaikymas Lietuvos rinkoje;
- Programinės įrangos, skirtos duomenų nuskaitymui, apdorojimui ir analizei, sukūrimas.

Teorinėje darbo dalyje (1 skyrius) pateikiama visa su analize susijusi teorija: efektyvios rinkos hipotezė, Roll mato išvedimas ir savybės, aprašomas duomenų stacionarumo tyrimo metodas, panaudojant RA-kriterijų (*reverse arrangement test*). Tiriamojoje dalyje (2 skyrius) pateikiami sukurti algoritmai, skirti viešai skelbiamų duomenų paruošimui ir analizei, bei gauti analizės rezultatai. Pastarieji aptariami diskusijos skyriuje. Sukurtos programinės įrangos aprašymas yra 3-ame skyriuje.

[1] Pavadintas straipsnio autoriaus pavarde.

1. TEORINĖ DALIS

1.1. EFEKTYVIOS RINKOS HIPOTEZĖ

Efektyvios rinkos hipotezė (*efficient market hypothesis, EMH*) yra idėja, kad finansų rinkos yra efektyvios, t.y. vertybinių popierių kainos jau atspindi tikrąją ekonominę vertę, kuri yra pagrįsta faktine informacija. Pagal šią teoriją, visa svarbiausia informacija apie investicijos objektą yra sukoncentruota to instrumento kainoje. Jeigu rinka yra efektyvi, tai instrumento rinkos kaina greitai reaguoja ir koreguojasi pagal naujai gaunamą informaciją ([3]).

Yra skiriamos trys efektyvios rinkos formos:

1. Silpna forma (*weak form*) yra efektyvios rinkos teorijos dalis, sakanti, kad rinkos (arba ją reprezentuojančios vertybinių popierių biržos, arba tarpbankinės rinkos) kainos atspindi visus istorinius kainų apimties duomenis, t.y. rinkos duomenis. Kitaip tariant, rinkos duomenys jau yra paveikę esamą kainą, todėl neturi jokios reikšmės ateities kainos pokyčių spėjimui. Jei pasitvirtina, kad efektyvios rinkos forma yra silpna, tai praeities kainų pokyčiai nebus susiję su ateities kainų pokyčiais.
2. Pusiau stipri forma (*semi strong form*) – efektyvios rinkos teorijos dalis, sakanti, kad rinkos kainos atspindi ne tik istorinius kainų ir apimties duomenis, bet ir visą viešai prieinamą informaciją – emitento pelną, dividendus, informaciją apie akcijų dalijimą, naujus produktus, finansavimo problemas, apskaitos pokyčius, likvidumą ir kt. Pusiau stipri forma apima silpną formą, nes jai priklauso rinkos duomenys plius vieša informacija. Investuotojai, besiremdami informacija, kai ji tapo vieša, tokioje rinkoje negalės gauti didesnių negu vidutiniai pelnų, nes kaina jau ėmė atspindėti naują viešą informaciją.
3. Stipri forma (*strong form*) – efektyvios rinkos teorijos dalis, sakanti, kad rinkos kainos atspindi visą informaciją, tiek viešą, tiek neviešą (t.y. apibrėžtą tam tikroje grupėje, pvz. firmos darbuotojų). Nevieša informacija prieinama tik su emitentu susijusiems asmenims (direktoriams, tarnautojams, daugiau kaip 10 proc. akcijų turintiems akcininkams) ir kai kada rinkos profesionalams. Jei efektyvios rinkos yra stiprios formos, tai nė viena investuotojų grupė negali per numatytą laiko tarpą gauti viršpelnio, pasinaudodama kainai darančia įtaką informacija kaip pranašumu.

   Kad egzistuotų informacijos požiūriu efektyvi rinka, reikalinga:

   - Didelis skaičius nepriklausomai analizes darančių ir pelno maksimizavimo siekiančių dalyvių;
   - Nauja informacija į rinką turi patekti atsitiktine tvarka;
   - Konkuruojantys investuotojai greitai keičia rinkos kainą, atsižvelgdami į gautą informaciją.

1.2. PRAŠOMOS-SIŪLOMOS KAINOS EFEKTYVAUS SKIRTUMO MATAS AKCIJŲ RINKOJE (ROLL[2] MATAS)

Geriausios prašomos-siūlomos kainos skirtumas (*quoted spread/(inside) bid-ask spread*)[3] naudojamas prekybos kaštams (*trading/transaction costs*) ir rinkos efektyvumui vertinti. Taip pat, tai yra matas, parodantis, kokią didžiausią kainą (papildomai) investuotojas yra pasiruošęs papildomai sumokėti pirkdamas ir parduodamas akciją per trumpą laiko tarpą (t.y. už *round-trip* sandorį). Šis skirtumas apskaičiuojamas:

$$paprastas\ kainų\ skirtumas_t = A_t - B_t; \qquad \textbf{(1.1)}$$

čia A_t - geriausia pardavimo kaina (*inside ask price*) momentu *t*,

B_t - geriausia pirkimo kaina (*inside bid price*) momentu *t*.

Prekyboje sandoriai yra vykdomi ne vien tik geriausiomis pirkimo ar pardavimo kainomis, bet ir kainomis, patenkančiomis į arba už paklausos ir pasiūlos ribų[4] [6]. Tokiu atveju, vietoj paprasto kainų skirtumo naudojamas efektyvus prašomos-siūlomos kainos skirtumas[5] (*effective bid-ask spread*):

$$efektyvus\ kainų\ skirtumas_t = 2 \cdot \left| P_t - \frac{B_t + A_t}{2} \right|; \qquad \textbf{(1.2)}$$

čia P_t - laiko momentu *t* įvykusio sandorio kaina,

B_t - geriausia pirkimo kaina, kai momentu t vyko sandoris,

A_t - geriausia pardavimo kaina, kai momentu t vyko sandoris.

Tiek paprasto, tiek efektyvaus kainų skirtumo apskaičiavimui, kiekvienu laiko momentu *t*, reikalinga informacija apie geriausius pavedimus pirkti ir parduoti. Ne visos biržos viešai teikia tokią informaciją. O jeigu ji ir prieinama, tai tokios informacijos kaupimas reikalauja daug laiko ir resursų sąnaudų. Metodas, aprašytas [5] straipsnyje, leidžia apskaičiuoti akcijos efektyvų kainų skirtumą, naudojant tiktai turimas rinkos kainas.

Mato išvedimas remiasi dvejomis prielaidomis:

1. Akcija prekiaujama efektyvioje rinkoje;

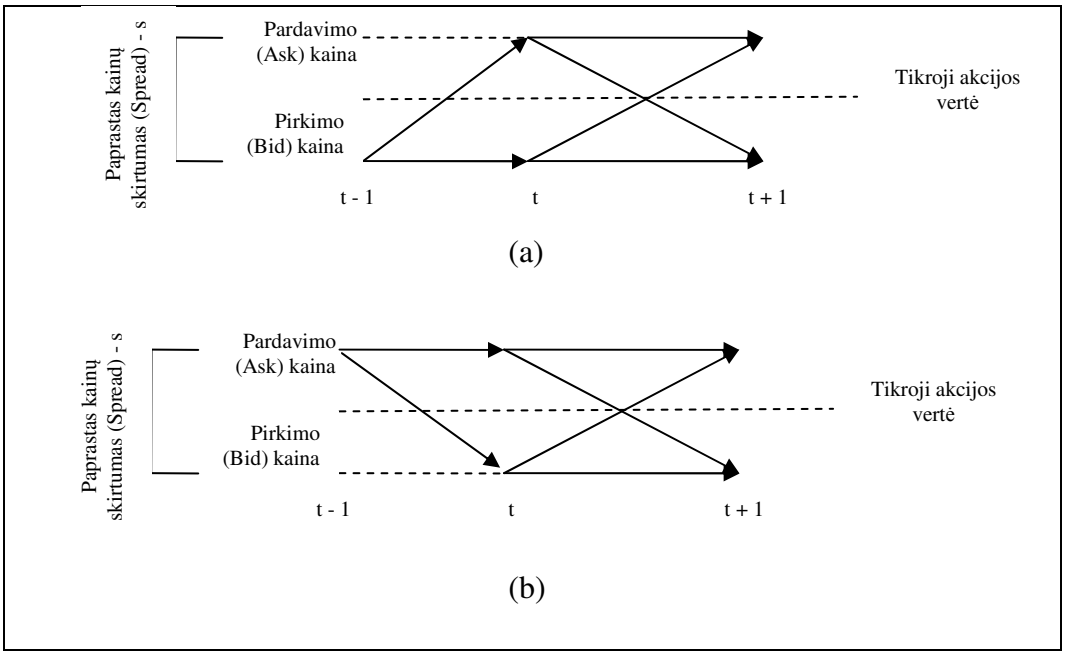
---

[2] Pavadintas [5] straipsnio autoriaus pavarde.
[3] Toliau tekste paprastas kainų skirtumas.
[4] Paklausos ir pasiūlos ribos – kainos, patenkančios į skirtumą tarp pavedimo pirkti didžiausia kaina ir pavedimo parduoti mažiausia kaina.
[5] Toliau tekste efektyvus kainų skirtumas.

2. Akcijos rinkos kainų skirtumų tikimybių skirstinys yra stacionarus (nekintantis) (bent jau nedideliam laiko intervalui).

Paprastumo dėlei, nagrinėkime situaciją, kada visų sandorių vienas iš dalyvių yra rinkos formuotojas (*market maker*). Tada prekiautojai (*traders*) akcijas perka tik iš jo ir tik jam jas parduoda. Be to, rinkos formuotojas visą laiką paprastą kainų skirtumą laiko pastovų (tarkim *s* litų). Efektyvioje rinkoje šis kainų skirtumas apgaubia tikrąją akcijos vertę (tarkim, kad ta vertė yra paprasto kainų skirtumo vidurys). Jei rinkoje apie akciją nėra jokios naujos informacijos, tikslinga manyti, kad tikimybės rinkos formuotojui pirkti ar parduoti akciją, kuomet prekiautojai atsitiktinai ateina dėl savų priežasčių, yra vienodos.

1.1 pav. pavaizduoti visi galimi akcijos rinkos kainos judėjimo keliai laiko momentais t ir $t + 1$, kai rinkoje apie šią akciją nebuvo jokios naujos informacijos. (a) – kai laiko momentu $t - 1$ akcija buvo parduota rinkos formuotojui. (b) – kai laiko momentu $t - 1$ akcija buvo nupirkta iš rinkos formuotojo. Visi keliai yra vienodai galimi.



1.1 pav. Galimi akcijos rinkos kainos judėjimo keliai

Tokiu būdu, akcijos rinkos kainų skirtumų Δp_t ir Δp_{t+1} $\left(\Delta p_t = p_t - p_{t-1}; \Delta p_{t+1} = p_{t+1} - p_t\right)$ sankirtos tikimybė priklauso nuo to, ar prieš tai sandoris (įvykęs ne dėl rinkoje atsiradusios naujos informacijos apie akciją, o dėl kitų priežasčių) įvyko pardavimo (*ask*), ar pirkimo (*bid*) kaina (1.1 lentelė).

1.1 lentelė

Akcijos rinkos kainų skirtumų sankirtos tikimybių skirstiniai

		Δp_t	
		0	+s
Δp_{t+1}	-s	0	1/4
	0	1/4	1/4
	+s	1/4	0

(a) p_{t-1} – pirkimo kaina

		Δp_t	
		-s	0
Δp_{t+1}	-s	0	1/4
	0	1/4	1/4
	+s	1/4	0

(b) p_{t-1} – pardavimo kaina

Kadangi P(sandorio kaina laiko momentu t – 1 yra pirkimo kaina) = P(sandorio kaina laiko momentu t – 1 yra pardavimo kaina), tai bendras sankirtos skirstinys yra:

1.2 lentelė

Akcijos rinkos kainų skirtumų bendras sankirtos tikimybių skirstinys

		Δp_t		
		-s	0	+s
Δp_{t+1}	-s	0	1/8	1/8
	0	1/8	1/4	1/8
	+s	1/8	1/8	0

Tuomet kovariacija, tarp vienas paskui kitą einančių akcijos rinkos kainų skirtumų, yra:

$$\operatorname{cov}(\Delta p_t, \Delta p_{t+1}) = E\big(\Delta p_t - E(\Delta p_t)\big)\big(\Delta p_{t+1} - E(\Delta p_{t+1})\big) = E(\Delta p_t \Delta p_{t+1}) = -\frac{1}{8}s^2 - \frac{1}{8}s^2 = -\frac{1}{4}s^2.$$

Taigi,

$$s = 2\sqrt{-\operatorname{cov}(\Delta p_t, \Delta p_{t+1})}\,; \qquad \textbf{(1.3)}$$

čia s nebūtinai yra paprastas kainų skirtumas (*quoted spread*). s taip pat gali būti ir efektyvus kainų skirtumas (*effective spread*).

1.3. ROLL MATO SAVYBĖS

1. Jei rinka yra efektyvi, tuomet kovariacija, tarp vienas paskui kitą einančių akcijos rinkos kainų skirtumų $(\Delta p_1, \Delta p_2, \ldots)$, atsiranda ne dėl naujos informacijos.

Esant efektyviai rinkai, efektyvus kainų skirtumas apgaubia tikrąją akcijos vertę. Tarkim tai yra tiesa ir šią akcijos vertę pažymėkime p^*. Tuomet laiko momentu t išmatuotas (žinomas) akcijos rinkos kainų skirtumas susideda iš dviejų komponentų: p^* pokyčio, kuris atsiranda dėl naujos informacijos

rinkoje, ir komponento, kuris visiškai apibrėžiamas tuo, ar sandoris laiko momentu *t* buvo inicijuotas tos pačios pusės dalyvio (pirkėjo ar pardavėjo), ar ne. Taigi, išmatuotas akcijos rinkos kainų skirtumas $\Delta\hat{p}_t$ yra aprašomas lygybe:

$$\Delta\hat{p}_t = \Delta p_t^* + \Delta p_t .$$

Jeigu rinka efektyvi, tuomet

$$\text{cov}(\Delta p_t^*, \Delta p_{t-j}^*) = 0 \qquad j \neq 0 \tag{1.4}$$

Taip pat reikalaujame, kad

$$\text{cov}(\Delta p_t^*, \Delta p_{t-j}) = 0 \tag{1.5}$$

(1.4) lygybę grindžiame tuo, kad efektyvioje rinkoje tikrosios akcijos vertės pokyčiai yra neprognozuojami (netikėti), o (1.5) – tuo, kad akcijos kainos šuoliai, tarp geriausių pirkimo (*bid*) ir pardavimo (*ask*) kainų, negali būti atspėti naudojantis tikrosios akcijos vertės pokyčiais, bei šie šuoliai negali būti naudojami tikrosios akcijos vertės pokyčių spėjimui.

Pasinaudojus (1.4) ir (1.5) gauname:

$$\begin{aligned}\text{cov}(\Delta\hat{p}_t, \Delta\hat{p}_{t-1}) &= \text{cov}(\Delta p_t^* + \Delta p_t, \Delta p_{t-1}^* + \Delta p_{t-1}) \\ &= \text{cov}(\Delta p_t^*, \Delta p_{t-1}^*) + \text{cov}(\Delta p_t^*, \Delta p_{t-1}) + \text{cov}(\Delta p_t, \Delta p_{t-1}^*) + \text{cov}(\Delta p_t, \Delta p_{t-1}) \\ &= \text{cov}(\Delta p_t, \Delta p_{t-1}) = \frac{-s^2}{4}.\end{aligned}$$

Taigi, kovariacija, tarp vienas paskui kitą einančių akcijos rinkos kainų skirtumų, atsiranda ne dėl naujos informacijos.

2. Jeigu rinka yra efektyvi, tuomet Roll matas nepriklauso nuo akcijos rinkos kainų stebėjimo intervalo. Tai yra, nepriklauso ar akcijos rinkos kainų pokyčiai skaičiuojami kas diena, ar kas mėnesį, ar kitu laikotarpiu.

Kovariaciją skaičiuokime ilgesniam laikotarpiui. Šio laikotarpio (tarkim iš *N* periodų) akcijos rinkos kainų pokytis yra suma, vienas paskui kitą einančių, *N* rinkos kainų pokyčių:

$$\Delta\hat{p}_T = \sum_{t=(T-1)N+1}^{T\cdot N} \Delta\hat{p}_t \ ;$$

čia T ilgesnio intervalo indeksas.

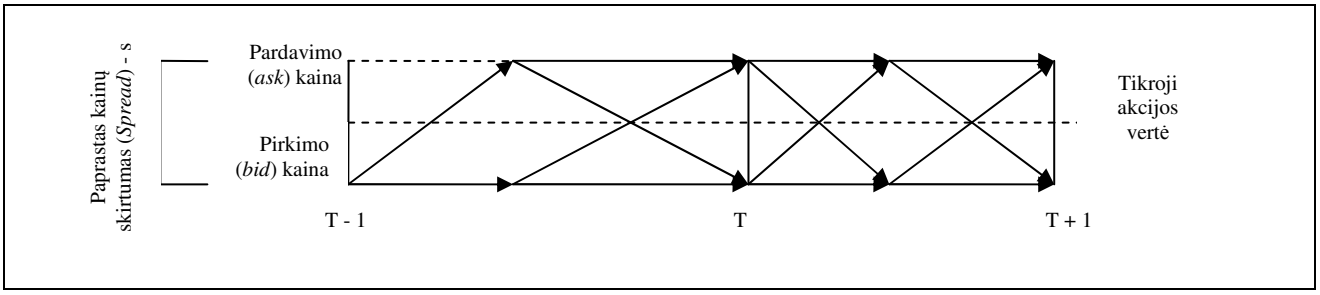
Tuomet,

$$\Delta\hat{p}_T = \Delta p_T^* + \Delta p_T ;$$

čia $\Delta p_T^* = \sum_{t=(T-1)N+1}^{T\cdot N} \Delta p_t^*$ ir $\Delta p_T = \sum_{t=(T-1)N+1}^{T\cdot N} \Delta p_t$.

Komponentų Δp_T ir Δp_{T+1} sankirtos tikimybių skirstinys yra toks pats kaip ir Δp_t bei Δp_{t+1}. Nors šiuo atveju kaina šokinėja pirmyn ir atgal tarp geriausių pirkimo ir pardavimo kainų (maksimaliai *N* kartų intervale nuo $T-1$ iki T), tačiau ji visada įgyja vieną iš šių dviejų.

Kai $N=2$ ir laiko momentu $T-1$ akcijos rinkos kaina buvo geriausia pirkimo kaina, kainos kitimo schema pavaizduota **1.2** pav.



1.2 pav. Akcijos rinkos kainos kitimo schema, kai vienas periodas padalintas į dvi dalis

Bendruoju atveju, tarp laiko momentų $T-1$ ir T (kai momentu $T-1$ akcijos rinkos kaina buvo geriausia pirkimo kaina) yra galimi 2^N akcijos kainos kitimo keliai ir 2^{N+1} galimų kelių, laikotarpiu nuo T iki $T+1$. Visi keliai yra vienodai galimi, bet kaip matome iš **1.2** pav. lygiai su puse iš jų gauname tokią pačią Δp reikšmę. Taigi, $\operatorname{cov}(\Delta p_T, \Delta p_{T-1}) = \dfrac{-s^2}{4}$.

Kai laiko intervalai nepersidengia, sumoms taikome (1.4) ir (1.5). Tokiu būdu, $\forall j$ gauname

$$\operatorname{cov}(\Delta p_T^*, \Delta p_{T-j}^*) = \operatorname{cov}(\Delta p_T^*, \Delta p_{T-j}) = 0.$$

Taigi,

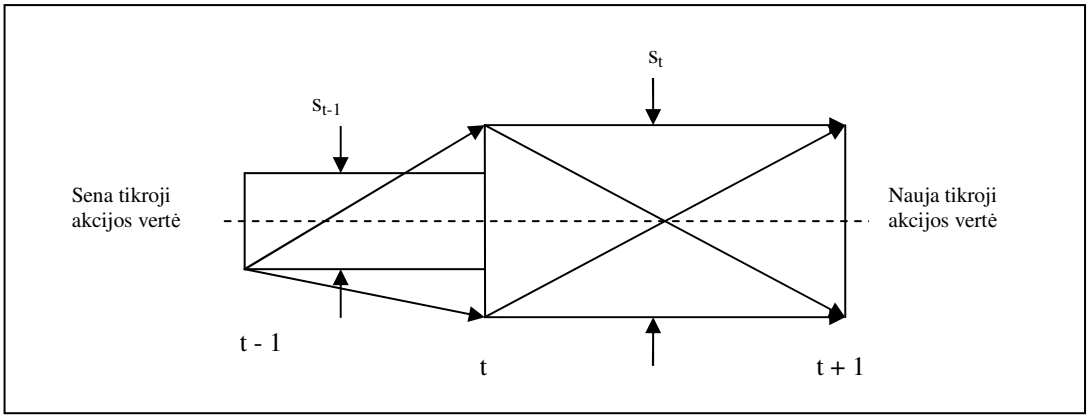
$$\operatorname{cov}(\Delta \hat{p}_T, \Delta \hat{p}_{T-1}) = \frac{-s^2}{4}$$

ir nuo *N* nepriklauso.

3. Net jeigu paprastas kainų skirtumas keičiasi priklausomai nuo prekiautojų reakcijos į naujienas, kovariacija vis tiek išliks $\dfrac{-s^2}{4}$, tik čia s^2 - imties geriausių pirkimo-pardavimo kainų skirtumų, pakeltų antruoju laipsniu, vidurkis.

Tarkime, kad paprastas kainų skirtumas reaguoja į naujos informacijos patekimą į rinką. Būtų protinga manyti, kad šis skirtumas gali padidėti daugiau negu absoliutus kainos pokytis tarp $t-1$ ir t. Taip pat, kad šis padidėjimas galėtų įvykti nepaisant to ar informacija, kuri iššaukė kainos pokytį, yra teisinga ar bloga. Jeigu skirtumas reaguoja simetriškai, tai neprarandant bendrumo 1.3 paveiksle pateikta schema gali būti naudojama proceso modeliavimui.

Rinkoje atsiradus naujai informacijai, akcijos tikroji vertė pasikeitė. Taip pat pasikeitė ir paprastas kainų skirtumas, todėl galima apibrėžti pokytį: $\Delta s = s_t - s_{t-1}$ (jis gali būti tiek teigiamas, tiek neigiamas).



1.3 pav. Galimi akcijos rinkos kainos judėjimo keliai, kai paprastas kainų skirtumas kinta laike

Dėl simetriškumo prielaidos (pirkimo-pardavimo kainų skirtumų pokytis Δs nepriklauso nuo algebrinio tikrosios akcijos vertės pokyčio ženklo) $\operatorname{cov}(\Delta p_{t-1}^*, \Delta p_t) = \operatorname{cov}(\Delta p_t^*, \Delta p_{t-1}) = 0$. Dėl rinkos efektyvumo $\operatorname{cov}(\Delta p_t^*, \Delta p_{t-1}^*) = 0$. Šiuo atveju akcijos rinkos kainų skirtumų sankirtos skirstinys yra:

1.3 lentelė

Akcijos rinkos kainų skirtumų sankirtos skirstinys, kai paprastas kainų skirtumas kinta laike

Δp_{t+1} \ Δp_t	$-s_{t-1}-\Delta s/2$	$-\Delta s/2$	$\Delta s/2$	$s_{t-1}+\Delta s/2$
$-s_t$	0	0	1/8	1/8
0	1/8	1/8	1/8	1/8
s_t	1/8	1/8	0	0

Atlikus skaičiavimus gauname:

$$\operatorname{cov}(\Delta p_t, \Delta p_{t+1}) = -\frac{1}{4} s_t^2.$$

Vidutinė kovariacijos reikšmė visoje aibėje:

$$E\big(\operatorname{cov}(\Delta p_t, \Delta p_{t+1})\big) = -\frac{1}{4\tau}\sum_t s_t^2 = -\frac{1}{4}\overline{s^2},\ t = 1,\ldots,\tau\,.$$

1.4. PROCESO STACIONARUMAS

1.4.1. BENDROSIOS SĄVOKOS

Sakykime, kad turime atsitiktinį procesą $\{X(t), t \in T\}$.

Procesas vadinamas stacionariuoju siaurąja prasme, jei jo baigtiniamačiai skirstiniai nepriklauso nuo laiko momentų postūmio. Tai reiškia, kad su visais $t_k \in T$ ir $t_k + \tau \in T$ $(k = \overline{1,n})$ n-matė pasiskirstymo funkcija

$$F(x_1, x_2, \ldots, x_n \mid t_1, t_2, \ldots, t_n) = F(x_1, x_2, \ldots, x_n \mid t_1 + \tau, t_2 + \tau, \ldots, t_n + \tau).$$

Procesas vadinamas stacionariuoju plačiąja prasme, jei

- $EX(t) = \mu$ (vidurkis pastovus);
- $C_{xx}(t, t+\tau) = E(X(t) - \mu)(X(t+\tau) - \mu) = C_{xx}(\tau, 0) = C_{xx}(\tau)$ (autokovariacinė funkcija priklauso tik nuo argumentų skirtumo).

Autokoreliacinė funkcija apibrėžiama kaip: $R_{xx}(\tau) = E(X(t)X(t+\tau))$. Būtina ir pakankama sąlyga, kad $R_{xx}(\tau)$ būtų autokoreliacinė, stacionaraus plačiąja prasme, proceso funkcija yra $R_{xx}(\tau) = R_{xx}(-\tau)$ ir $R_{xx}(\tau)$ neneigiama apibrėžta funkcija.

Kai procesas yra stacionarus plačiąja prasme autokovariacinė ir autokoreliacinė funkcijos siejasi lygybe:

$$C_{xx}(\tau) = R_{xx}(\tau) - \mu^2.$$

1.4.2. DUOMENŲ STACIONARUMO TYRIMAS

Norint patvirtinti duomenų stacionarumą, teoriškai reiktų patvirtinti, kad visos statistinės savybės yra nekintančios laike [2]. Kadangi galimų statistikų skaičius yra begalinis, tai praktikoje toks patvirtinimas yra neįmanomas. Tokiu atveju, tiriant duomenų stacionarumą, tenka priimti keletą prielaidų:

1. Jei duomenys yra stacionarūs, tuomet, kiekviename trumpame laiko intervale suskaičiuoti, statistinių savybių vidurkiai vienas nuo kito statistiškai reikšmingai nesiskirs.
2. Tiriama duomenų aibė yra pakankamai didelė ir reprezentatyvi.

3. Stacionarumo plačiąja prasme patvirtinimas yra pakankama stacionarumo sąlyga tolimesniems skaičiavimams ir analizei.
4. Jeigu proceso vidutinė kvadratinė reikšmė yra stacionari, tuomet autokoreliacinė funkcija taip pat yra stacionari. Šią prielaidą galima aiškinti tuo, kad nestacionariems duomenims, kurių autokoreliacinė funkcija keičiasi laike, nebūdinga išlaikyti nekintančią šios funkcijos reikšmę ir kai $\tau = 0$ ([1]). Kadangi $R_{xx}(0) = EX^2(t)$, tai dėmesys gali būti sukoncentruotas į vidutinę kvadratinę reikšme, o ne į visą autokoreliacinę funkciją.

Nepamirštant ankščiau išvardintų prielaidų, proceso $X(t)$ stacionarumą galima tirti tokiu būdu ([1]):

Visas laiko intervalas padalinamas į *N*, lygių laiko atžvilgiu, dalių. Duomenys kiekviename intervale turi būti nepriklausomi nuo duomenų esančių kitame (po to einančiame) intervale.

Kiekviename intervale skaičiuojame vidurkį ir vidutinį kvadratinį dydį:

$$\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_N,$$

$$\overline{x_1^2}, \overline{x_2^2}, \ldots, \overline{x_N^2}.$$

Tiriame, ar vidurkių ir vidutinių kvadratinių dydžių sekose yra koks nors trendas arba svyravimai, atsiradę ne dėl imties atsitiktinumo. Šiai užduočiai tinka neparametrinis RA (*reverse arrangement test*) kriterijus, kuris proceso stacionarumui tirti, taikomas tokiu būdu:

Tegul H0: $\overline{x_1^2}, \overline{x_2^2}, \ldots, \overline{x_N^2}$ seka reprezentuoja nepriklausomų, atsitiktinio dydžio *X*, kurio vidutinė kvadratinė reikšmė ψ^2, matavimų seką. O $\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_N$ seka – nepriklausomų, to paties atsitiktinio dydžio *X*, kurio vidurkis μ, matavimų seką.

Jei ši hipotezė teisinga, svyravimai sekose bus atsitiktiniai ir nebus jokio trendo. Tokiu būdu, procesas stacionarus. Jei hipotezė neteisinga, priimame, kad procesas nėra stacionarus.

1.4.3. RA KRITERIJUS

„Reverse arrangement test“ – neparametrinis kriterijus, naudojamas monotoninio trendo aptikimui. Jis yra paprastas ir nereikalauja jokių žinių apie atsitiktinio dydžio skirstinį.

Tarkim turim atsitiktinio dydžio *X* imtį iš *N* elementų: $(x_1, x_2, \ldots, x_N)$. Aibės $x_1, x_2, \ldots, x_N$ elementams apibrėžkime dydį

$$h_{ij} = \begin{cases} 1, \text{ jei } x_i > x_j, \forall \text{i} < \text{j} \\ 0, \text{ priešingu atveju} \end{cases}.$$

Tuomet

$$A = \sum_{i=1}^{N-1} A_i \text{ , čia } A_i = \sum_{j=i+1}^{N} h_{ij} .$$

Jeigu $x_1, x_2, \ldots, x_N$ yra nepriklausomi, to paties atsitiktinio dydžio stebiniai, tuomet priešingų išdėstymų kiekis yra atsitiktinis dydis A, kurio vidurkis ir dispersija:

$$\mu_A = \frac{N(N-1)}{4},$$

$$\sigma_A^2 = \frac{2N^3 + 3N^2 - 5N}{72} = \frac{N(2N+5)(N-1)}{72}.$$

H0: stebiniai yra nepriklausomi ir to paties atsitiktinio dydžio X, bei nėra jokio trendo.

Nulinės hipotezės priėmimo sritis:

$$[A_{N;1-\alpha/2} < A \le A_{N;\alpha/2}];$$

čia α - reikšmingumo lygmuo.

Apytikslėms $A_{N;\alpha}$ reikšmėms, tokioms, kad $P(A_N > A_{N;\alpha}) = \alpha$, apskaičiuoti gali būti naudojama formulė ([4]):

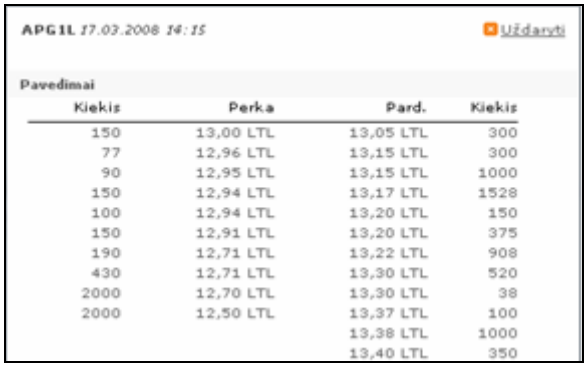
$$z = \frac{A_{N;\alpha} - \frac{N(N-1)}{4} + 0.5}{\sqrt{\frac{N(2N+5)(N-1)}{72}}};$$

čia z pasiskirstęs pagal standartinį normalųjį skirstinį.

2. TIRIAMOJI DALIS

2.1. DUOMENŲ AIBĖS SUDARYMAS

Vilniaus vertybinių popierių biržos (toliau VVPB) internetiniame puslapyje [8] yra skelbiama vieša listinguojamų akcijų informacija. Ji skirta ne verslo tikslams, o tam, kad visuomenė tikėtų prekybos skaidrumu. Dėl šios priežasties, duomenys minučių tikslumu yra teikiami realiu laiku su 15 min. pavėlinimu. Tokių duomenų nuskaitymui buvo sukurta programinė įranga, aprašyta 3.1.1. skyriuje. Su šia programa galima skaityti tiek tiesiogiai laike pateikiamus rinkos gylio lentelės (2.1 pav.), tiek istorinius, įvykusių sandorių (2.2 pav.) ir bendrosios informacijos (2.3 pav.), duomenis.

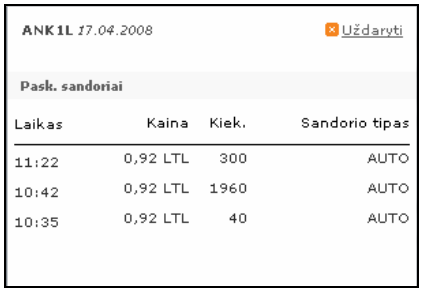


APG1L 17.03.2008 14:15 Uždaryti

Pavedimai

Kiekis	Perka	Pard.	Kiekis
150	13,00 LTL	13,05 LTL	300
77	12,96 LTL	13,15 LTL	300
90	12,95 LTL	13,15 LTL	1000
150	12,94 LTL	13,17 LTL	1528
100	12,94 LTL	13,20 LTL	150
150	12,91 LTL	13,20 LTL	375
190	12,71 LTL	13,22 LTL	908
430	12,71 LTL	13,30 LTL	520
2000	12,70 LTL	13,30 LTL	38
2000	12,50 LTL	13,37 LTL	100
		13,38 LTL	1000
		13,40 LTL	350

2.1 pav. VVPB pateikiama akcijos rinkos gylio lentelė

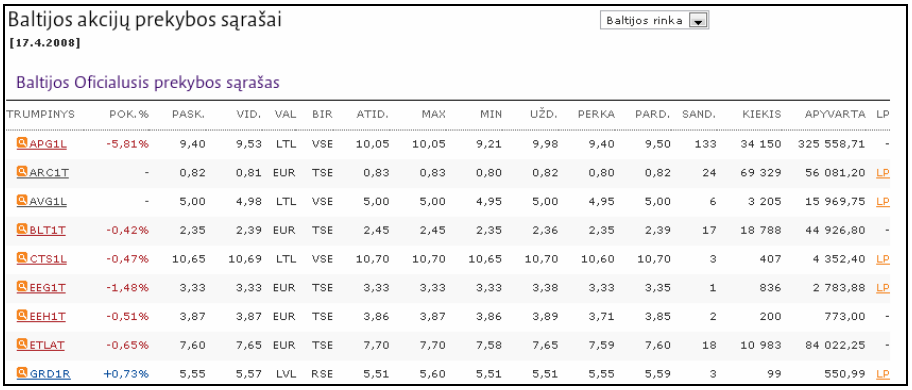


ANK1L 17.04.2008 Uždaryti

Pask. sandoriai

Laikas	Kaina	Kiek.	Sandorio tipas
11:22	0,92 LTL	300	AUTO
10:42	0,92 LTL	1960	AUTO
10:35	0,92 LTL	40	AUTO

2.2 pav. VVPB pateikiama akcijos įvykusių sandorių duomenų suvestinė



Baltijos akcijų prekybos sąrašai [17.4.2008] Baltijos rinka

Baltijos Oficialusis prekybos sąrašas

TRUMPINYS	POK.%	PASK.	VID.	VAL	BIR	ATID.	MAX	MIN	UŽD.	PERKA	PARD.	SAND.	KIEKIS	APYVARTA	LP
APG1L	-5,81%	9,40	9,53	LTL	VSE	10,05	10,05	9,21	9,98	9,40	9,50	133	34 150	325 558,71	-
ARC1T	-	0,82	0,81	EUR	TSE	0,83	0,83	0,80	0,82	0,80	0,82	24	69 329	56 081,20	LP
AVG1L	-	5,00	4,98	LTL	VSE	5,00	5,00	4,95	5,00	4,95	5,00	6	3 205	15 969,75	LP
BLT1T	-0,42%	2,35	2,39	EUR	TSE	2,45	2,45	2,35	2,36	2,35	2,39	17	18 788	44 926,80	-
CTS1L	-0,47%	10,65	10,69	LTL	VSE	10,70	10,70	10,65	10,70	10,60	10,70	3	407	4 352,40	LP
EEG1T	-1,48%	3,33	3,33	EUR	TSE	3,33	3,33	3,33	3,38	3,33	3,35	1	836	2 783,88	LP
EEH1T	-0,51%	3,87	3,87	EUR	TSE	3,86	3,87	3,86	3,89	3,71	3,85	2	200	773,00	-
ETLAT	-0,65%	7,60	7,65	EUR	TSE	7,70	7,70	7,58	7,65	7,59	7,60	18	10 983	84 022,25	-
GRD1R	+0,73%	5,55	5,57	LVL	RSE	5,51	5,60	5,51	5,51	5,55	5,59	3	99	550,99	LP

2.3 pav. VVPB pateikiama bendra informacija apie akcijas

Kadangi biržos oficialiame ir papildomame sąrašuose yra įtraukta apie 100 akcijų, tai laiko intervalai, tarp vienos akcijos nuskaitytų dviejų gretimų (laiko atžvilgiu) rinkos gylio lentelės „nuotraukų“, yra apie 1min. 30 s[6]. (2.4 pav. pateikta, kaip atrodo į duomenų bazę įrašyta rinkos gylio lentelės fotografija.) Siekiant sumažinti šiuos intervalus, analizei buvo pasirinktos 9 akcijos (2.1 lentelė). Visomis pasirinktomis akcijomis prekiaujama VVPB, dalis jų priklauso oficialiam, dalis papildomam prekybos sąrašui. Tam, kad visos 9 akcijos nebūtų vien likvidžios, arba vien nelikvidžios, renkantis buvo atsižvelgiama ir į per dieną įvykusių sandorių skaičių bei vertę. Tokiu būdu sumažinus analizuojamų akcijų skaičių, laiko intervalas, tarp vienos akcijos gretimų rinkos gylio lentelių nuotraukų, tapo 13 – 17 s.

DATA	TRUMPINYS	LAIKAS	KOMP_LAIK	PIRK_KIEK	PIRK_KAINA	PIRK_VAL	PARD_KAINA	PARD_VAL	PARD_KIEK
2008.03.17	APG1L	10:00	10:15:06	75	13,17	LTL	13,44	LTL	75
2008.03.17	APG1L	10:00	10:15:06	101	13,16	LTL	13,45	LTL	15
2008.03.17	APG1L	10:00	10:15:06	100	13,15	LTL	13,45	LTL	100
2008.03.17	APG1L	10:00	10:15:06	100	13,11	LTL	13,5	LTL	280
2008.03.17	APG1L	10:00	10:15:06	130	13,02	LTL	13,5	LTL	2000
2008.03.17	APG1L	10:00	10:15:06	250	13,01	LTL	13,5	LTL	300
2008.03.17	APG1L	10:00	10:15:06	2000	13	LTL	13,5	LTL	4000
2008.03.17	APG1L	10:00	10:15:06	2000	12,7	LTL	13,65	LTL	100
2008.03.17	APG1L	10:00	10:15:06	2000	12,5	LTL	13,75	LTL	2000
2008.03.17	APG1L	10:00	10:15:06	0	0		13,86	LTL	784
2008.03.17	APG1L	10:00	10:15:06	0	0		14	LTL	100

2.4 pav. Pradinė akcijos rinkos gylio lentelė

2.1 lentelė

Pasirinktų akcijų sąrašas

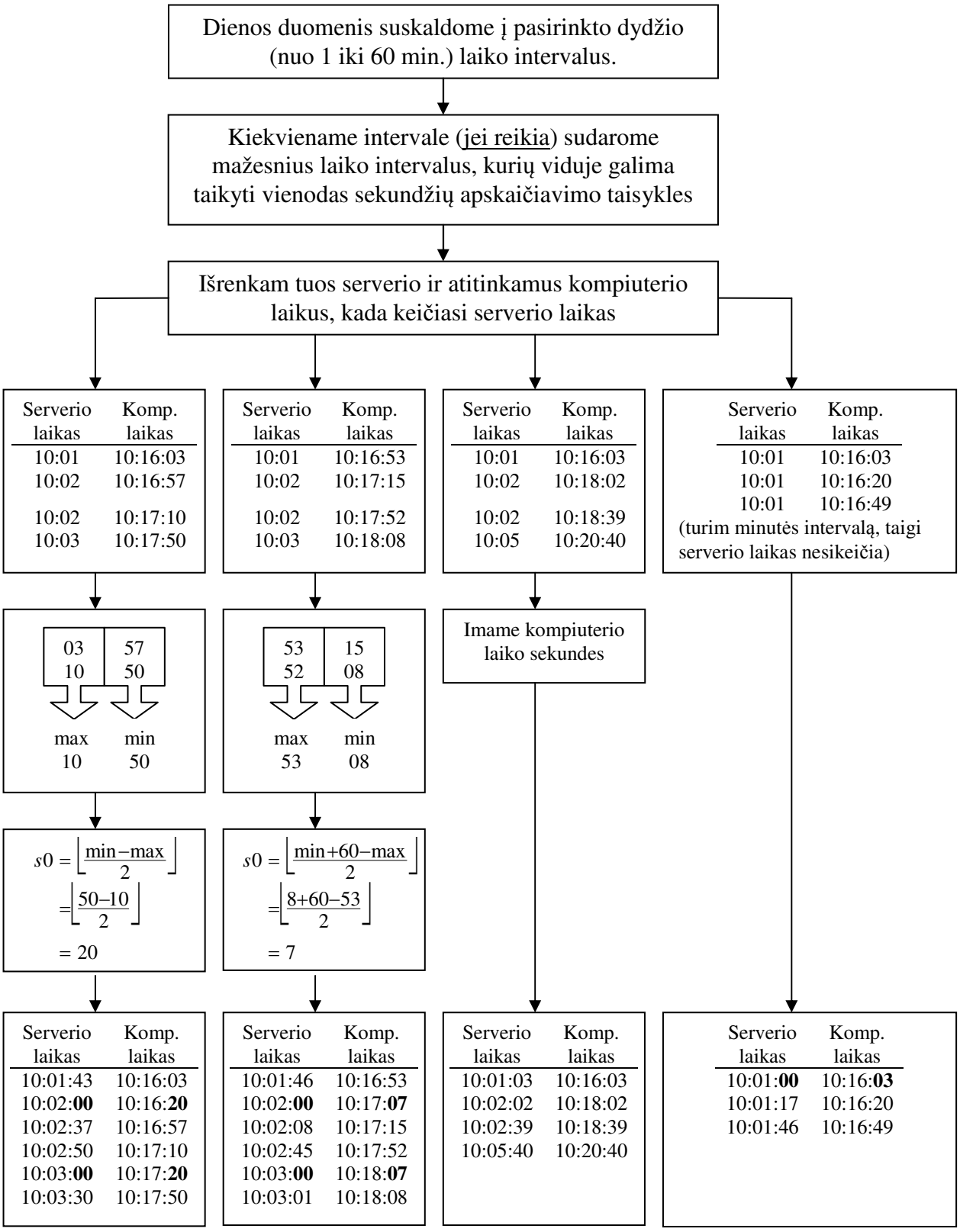
Akcijos trumpinys	Pilnas akcijos pavadinimas	Akcijos didžiausia kaina 2008 m. (iki 0509), Lt	Apyvarta 2008 m. (iki 0509), mln. Lt
ANK1L	Anykščių vynas	1,17	0,07
APG1L	Apranga	14,60	24,61
CTS1L	City Service AB	13,40	7,14
GRG1L	Grigiškės	2,70	0,53
IVL1L	Invalda	17,34	16,83
LDJ1L	Lietuvos dujos	3,65	3,38
NDL1L	DnB NORD bankas	455	1,83
SAN1L	Sanitas	29,41	8,38
TEO1L	Teo Lt	2,39	105,1

	Oficialus prekybos sąrašas
	Papildomas prekybos sąrašas

Stebint duomenų nuskaitymo procesą, pastebėta, kad VVPB serverio laiko ir kompiuterio, kuriame paleista duomenų nuskaitymo programa, laiko minutės keičiasi ne vienu metu. Dėl šios priežasties, buvo sukurtas algoritmas sekundžių, prie serverio laiko, prirašymui. Algoritme

[6] Šis laiko intervalas taip pat priklauso ir nuo interneto spartos.

skaičiavimams naudojome pradinės rinkos gylio lentelės (2.4 pav.) serverio laiką („Laikas“) ir kompiuterio laiką („Komp_laik“). Kiekvienai dienai ir kiekvienai akcijai taikomo algoritmo schema:



Prirašius sekundes tikrinama, ar naujieji serverio laikai (su sekundėmis) didėja tuo pačiu metu kaip ir jų atitinkami kompiuterio laikai. (Dėl apvalinimo paklaidų ir skirtingo suskirstymo į intervalus, kartais sekundės yra įrašomos neteisingai.) Jeigu laikai kartu nedidėja, išvedamas pranešimas apie įvykusią klaidą.

2.5 pav. pavaizduota, kaip duomenų bazėje atrodo rinkos gylio lentelė su prie serverio laiko prirašytomis sekundėmis.

DATA	TRUMPINYS	LAIKAS	KOMP_LAIK	PIRK_KIEK	PIRK_KAINA	PIRK_VAL	PARD_KAINA	PARD_VAL	PARD_KIEK
2008.03.17	APG1L	10:00:06	10:15:06	75	13,17	LTL	13,44	LTL	75
2008.03.17	APG1L	10:00:06	10:15:06	101	13,16	LTL	13,45	LTL	15
2008.03.17	APG1L	10:00:06	10:15:06	100	13,15	LTL	13,45	LTL	100
2008.03.17	APG1L	10:00:06	10:15:06	100	13,11	LTL	13,5	LTL	280
2008.03.17	APG1L	10:00:06	10:15:06	130	13,02	LTL	13,5	LTL	2000
2008.03.17	APG1L	10:00:06	10:15:06	250	13,01	LTL	13,5	LTL	300
2008.03.17	APG1L	10:00:06	10:15:06	2000	13	LTL	13,5	LTL	4000
2008.03.17	APG1L	10:00:06	10:15:06	2000	12,7	LTL	13,65	LTL	100
2008.03.17	APG1L	10:00:06	10:15:06	2000	12,5	LTL	13,75	LTL	2000
2008.03.17	APG1L	10:00:06	10:15:06	0	0		13,86	LTL	784
2008.03.17	APG1L	10:00:06	10:15:06	0	0		14	LTL	100

2.5 pav. Rinkos gylio lentelė prirašius sekundes

Siekiant sumažinti duomenų bazėje esančių įrašų skaičių (kuo mažiau įrašų, tuo greičiau veikia programinė įranga) duomenys buvo „sutraukiami“. Jei dvi gretimos (laiko atžvilgiu) rinkos gylio lentelės nuotraukos yra vienodos (t.y. visi pirkimų ir pardavimų duomenys sutampa, o skiriasi tiktai nuotraukų nuskaitymo laikai) lentelės yra „sujungiamos“. Pavyzdžiui, tarkim turim dvi gretimas (laiko atžvilgiu) rinkos gylio lentelės nuotraukas padarytas 10:01:16 ir 10:02:04. Šių lentelių duomenys yra identiški. Tuomet daroma prielaida, kad nuo 10:01:16 iki 10:02:04 (imtinai) neįvyko jokių pakitimų ir rinkos gylio lentelė buvo tokia pati. Tokiu būdu „sutraukiant“ duomenis pavyko sumažint duomenų bazėje esančių įrašų skaičių (pvz. ANK1L nuo 9112 iki 19, TEO1L nuo 115242 iki 14138 (2008 03 13)). **2.6** paveiksle pateikiama, kaip atrodo rinkos gylio lentelė po duomenų „sutraukimo“.

DATA	TRUMPINYS	PR_LAIKAS	PB_LAIKAS	PIRK_KIEK	PIRK_KAINA	PIRK_VAL	PARD_KAINA	PARD_VAL	PARD_KIEK
2008.03.17	APG1L	10:00:06	10:00:16	75	13,17	LTL	13,44	LTL	75
2008.03.17	APG1L	10:00:06	10:00:16	101	13,16	LTL	13,45	LTL	15
2008.03.17	APG1L	10:00:06	10:00:16	100	13,15	LTL	13,45	LTL	100
2008.03.17	APG1L	10:00:06	10:00:16	100	13,11	LTL	13,5	LTL	280
2008.03.17	APG1L	10:00:06	10:00:16	130	13,02	LTL	13,5	LTL	2000
2008.03.17	APG1L	10:00:06	10:00:16	250	13,01	LTL	13,5	LTL	300
2008.03.17	APG1L	10:00:06	10:00:16	2000	13	LTL	13,5	LTL	4000
2008.03.17	APG1L	10:00:06	10:00:16	2000	12,7	LTL	13,65	LTL	100
2008.03.17	APG1L	10:00:06	10:00:16	2000	12,5	LTL	13,75	LTL	2000
2008.03.17	APG1L	10:00:06	10:00:16	0	0		13,86	LTL	784
2008.03.17	APG1L	10:00:06	10:00:16	0	0		14	LTL	100

2.6 pav. Rinkos gylio lentelė „sutraukus“ duomenis

Stengiantis atkurti kuo tikresnį rinkos gylio lentelių vaizdą, buvo formuojamos jų nuotraukos prieš įvykusį sandorį ir po jo. Duomenų bazėje rinkos gylio lentelės nuotrauka, tam tikru laiko momentu *t*, yra identifikuojama pagal pradžios laiką („Pr_laikas“) ir pabaigos laiką („Pb_laikas“).

Pažymėkime:

Įvykusio sandorio duomenys

- laikasS – sandorio laikas (kada įvyko sandoris);
- kainaS – sandorio kaina (perleidžiamos akcijos kaina);
- kiekisS – sandorio dydis (perleidžiamų akcijų skaičius);

Rinkos gylio lentelės nuotraukos (momentu *t*) „A“ duomenys

- kainaPi_A - geriausia pirkimų kaina (pavedimo pirkti didžiausia kaina);
- kiekisPi_A – akcijų skaičius, kurį norima nupirkti geriausia pirkimų kaina (pirkimų kiekių stulpelio pirmoje eilutėje esantis kiekis);
- kainaPa_A – geriausia pardavimų kaina (pavedimo parduoti mažiausia kaina), kiekisPa_A – akcijų skaičius, kurį norma parduoti geriausia pardavimu kaina (pardavimų kiekių stulpelio pirmoje eilutėje esantis skaičius).

Algoritmą aprašykime naudodamiesi pavyzdžiu[7]:

Tarkim turim įvykusio sandorio duomenis: 10:01 (laikasS) 3,45Lt (kainaS) 100 vnt. (kiekisS). Tikslas – nustatyti (jei nėra – sukurti), kaip atrodė rinkos gylio lentelė prieš šį sandorį ir po jo. Kadangi sandorio įvykdymo laikas pateikiamas tik minučių tikslumu, mums reikia išrinkti visas rinkos gylio lentelių nuotraukas, kurios buvo nuo 10:01:00 iki 10:01:59.

1. Tokių nuotraukų nėra (taip gali būti, nes duomenų bazėje esančias rinkos gylio lentelių nuotraukas skiria laiko intervalas)

   Imam artimiausią (laiko atžvilgiu) rinkos gylio lentelės nuotrauką, kuri buvo prieš 10:01:00 (šią nuotrauką pažymėkime „A“). Į duomenų bazę įrašoma nauja lentelė, kurios „Pr_laikas“ = „Pb_laikas“ = 10:01:00, o pirkimų ir pardavimų duomenys yra tokie patys kaip ir nuotraukos „A“. Naująją lentelę nagrinėjame 2 punkte aprašytu algoritmu.

2. Yra tik viena nuotrauka (pažymėkime šią nuotrauką „B“)

   Priklausomai nuo nuotraukos „B“ „Pr_laiko“ ir „Pb_laiko“ tolimesnė algoritmo eiga skirstoma:

   *2.1. „Pr_laikas“ ir „Pb_laikas“ atitinkamai yra 10:**00**:ab – 10:**01**:cd arba 10:**01**:ab – 10:**01**:cd.*

   Ieškome, kuriame nuotraukos „B“ kainų stulpelyje (pirkimų ar pardavimų) yra kainaS (3,45 Lt).

   2.1.1. Kainos nėra nei viename stulpelyje

   Skaičiuojami skirtumai: $s1 = kainaS - kainaPiB$ ir $s2 = kainaPaB - kainaS$. Formuojama rinkos gylio lentelė prieš sandorį: naujos lentelės „Pr_laikas“ = „Pb_laikas“ = „B“ lentelės „Pb_laikas“ + 1s. KainaS ir kiekisS įterpiami į tos

[7] Kiekvienai dienai, kiekvienos akcijos kiekvienai minutei algoritmas yra kartojamas.

rinkos gylio lentelės pusės (pirkimų ar pardavimų) pirmąją eilutę, kur apskaičiuotas kainų skirtumas (s1 ar s2) yra mažesnis. Jei s1 = s2, generuojam tolygiai atkarpoje [0,1] pasiskirsčiusį atsitiktinį dydį X. Jei $X \leq 0.5$ – įvykusio sandorio duomenis įterpiame į pardavimų pusės pirmąją eilutę, priešingu atveju – į pirkimų pusės pirmąją eilutę.

Rinkos gylio lentelė po sandorio: „Pr_ laikas“ = „Pb_laikas“ = nuotraukos „B“ „Pb_laikas“ + 2s, pirkimų ir pardavimų duomenys tokie patys kaip ir nuotraukos „B“.

2.1.2. *Kaina yra kažkuriame stulpelyje (tarkim pirkimų kainų stulpelyje)*[8]

2.1.2.1. *kainaS = kainaPiB* (t.y. kaina yra pirmoji pirkimo kainų stulpelyje)

2.1.2.1.1.1. *kiekisS = kiekisPiB*

Nuotrauka „B“ yra nuotrauka prieš sandorį. Nuotrauka po sandorio: „Pr_laikas“ = „Pb_laikas“ = nuotraukos „B“ „Pb_laikas“ + 1s. Pavedimų parduoti duomenys tokie patys kaip ir nuotraukos „B“. Pavedimų pirkti duomenys gaunami iš nuotraukos „B“ pirkimų duomenų išbraukus pirmąja eilutę.

2.1.2.1.1.2. *kiekisS < kiekisPiB*

Nuotrauka „B“ yra nuotrauka prieš sandorį. Nuotrauka po sandorio: „Pr_laikas“ = „Pb_laikas“ = nuotraukos „B“ „Pb_laikas“ + 1s. Pavedimų parduoti duomenys tokie patys kaip ir nuotraukos „B“. Pavedimų pirkti duomenys gaunami kiekįPiB pakeitus į $s = kiekisPiB - kiekisS$.

2.1.2.1.1.3. *kiekisS > kiekisPiB*

Formuojama nuotrauka prieš sandorį: „Pr_laikas“ = „Pb_laikas“ = nuotraukos „B“ „Pb_laikas“ + 1s. Pavedimų parduoti duomenys yra tokie patys kaip ir nuotraukos „B“. Pavedimų pirkti duomenys suformuojami iš nuotraukos „B“ pirkimų duomenų, į pirmąją vietą įterpiant įvykusio sandorio duomenis (kainą ir kiekį).

Nuotrauka po sandorio: „Pr_laikas“ = „Pb_laikas” = nuotraukos „B” „Pb_laikas” + 2s. Pavedimų pirkti ir parduoti duomenys tokie patys kaip ir nuotraukos „B“.

[8] Jei kainaS yra pardavimų kainų stulpelyje, tai tolimesnė algoritmo eiga yra analogiška, tiktai visa analizė atliekama su pavedimų parduoti duomenimis.

2.1.2.2. *kainaS ≠ kainaPiB* (t.y. kaina nėra pirmoji pirkimų kainų stulpelyje, tarkim ji yra trečia)

Sukuriamos dvi naujos nuotraukos:

- Pirma nuotrauka: „Pr_laikas“ = „Pb_laikas“ = nuotraukos „B“ „Pb_laikas“ + 1s. Pavedimų parduoti duomenys tokie patys kaip ir nuotraukos „B“. Pavedimų pirkti duomenys formuojami iš nuotraukos „B“ pirkimo duomenų išbraukiant pirmąją eilutę.
- Antra nuotrauka: „Pr_laikas“ = „Pb_laikas“ = nuotraukos „B“ „Pb_laikas“ + 2s. Pavedimų parduoti duomenys tokie patys kaip ir nuotraukos „B“. Pavedimų pirkti duomenys formuojami iš nuotraukos „B“ pirkimo duomenų išbraukiant pirmas dvi eilutes. Tokiu būdu antrojoje nuotraukoje geriausia pirkimų kaina tampa lygi įvykusio sandorio kainai ir šios nuotraukos atžvilgiu vykdome 2.1.2.1 algoritmo punktą.

*2.2. „Pr_laikas“ ir „Pb_laikas“ atitinkamai yra 10:**01**:ab – 10:**02**:cd.*

Formuojama nauja nuotrauka: „Pr_laikas“ = „Pb_laikas“ = 10:01:00. Pavedimų pirkti ir parduoti duomenys yra tokie patys kaip ir nuotraukos „B“. Naujai suformuotos nuotraukos atžvilgiu vykdomas 2.1 punkte aprašytas algoritmas.

*2.3. „Pr_laikas“ ir „Pb_laikas“ atitinkamai yra MN:**mn**:ab – KL:**kl**:cd (MN:**mn**:ab < 10:01:00 <KL:**kl**:cd).*

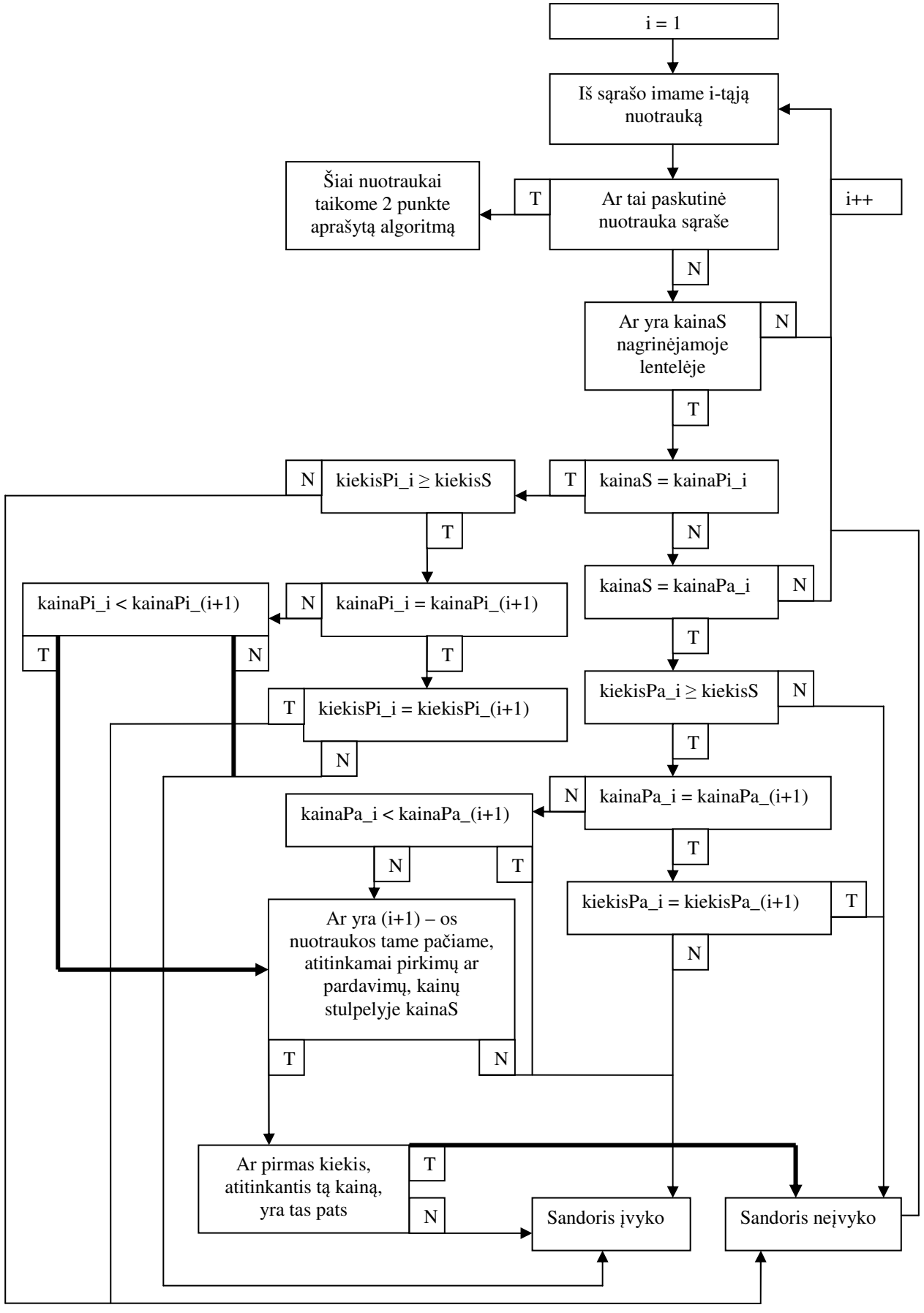
Vietoj nuotraukos „B“ sukuriamos trys naujos nuotraukos:

- Pirma: „Pr_laikas“ = MN:mn:ab, „Pb_laikas“ = 10:00:59. Pavedimų pirkti ir parduoti duomenys tokie patys kaip ir nuotraukos „B“.
- Antra: „Pr_laikas“ = „Pb_laikas“ = 10:01:00. Pavedimų pirkti ir parduoti duomenys tokie patys kaip ir nuotraukos „B“.
- Trečia: „Pr_laikas“ = 10:02:00, „Pb_laikas“ = KL:kl:cd. Pavedimų pirkti ir parduoti duomenys tokie patys kaip ir nuotraukos „B“.

Naujai suformuotos antrosios nuotraukos atžvilgiu taikome 2.1 punkte aprašytą algoritmą.

3. <u>Yra daugiau nei viena nuotrauka</u>

Iš šių nuotraukų sudarome, vienas paskui kitą (laiko atžvilgiu) einančių, nuotraukų sąrašą. Tuomet algoritmo schema:


i = 1
Iš sąrašo imame i-tąją nuotrauką
Ar tai paskutinė nuotrauka sąraše
T
Šiai nuotraukai taikome 2 punkte aprašytą algoritmą
N
i++
Ar yra kainaS nagrinėjamoje lentelėje
N
T
kainaS = kainaPi_i
T
kiekisPi_i ≥ kiekisS
N
T
kainaPi_i = kainaPi_(i+1)
N
kainaPi_i < kainaPi_(i+1)
T
N
T
kiekisPi_i = kiekisPi_(i+1)
N
N
kainaS = kainaPa_i
N
T
kiekisPa_i ≥ kiekisS
N
T
kainaPa_i = kainaPa_(i+1)
N
kainaPa_i < kainaPa_(i+1)
N
T
kiekisPa_i = kiekisPa_(i+1)
T
N
Ar yra (i+1) – os nuotraukos tame pačiame, atitinkamai pirkimų ar pardavimų, kainų stulpelyje kainaS
T
N
Ar pirmas kiekis, atitinkantis tą kainą, yra tas pats
T
N
Sandoris įvyko
Sandoris neįvyko

Jeigu sandoris įvyko, tai nagrinėjama lentelė i yra nuotrauka prieš sandorį. Nuotraukos po sandorio „Pr_laikas“ = „Pb_laikas“ = i-osios nuotraukos “Pb_laikas” + 1s. O pavedimų pirkti ar parduoti duomenys formuojami analogiškai kaip aprašyta 2-ame punkte, atsižvelgiant į sandorio iniciatorių ir dydį.

Prieš įrašant naują rinkos gylio lentelės nuotrauką į DB, tikrinama ar tinkami jos „Pr_laikas“ ir „Pb_laikas“. Atsižvelgiant į sandorio laiką, nuotraukų prieš ir po jo, „Pr_laikas“ ir „Pb_laikas“ turi būti tos pačios valandos ir minutės. Jei laikai blogi, išvedamas įvykusios klaidos pranešimas.

Kai sandorių, įvykusių tą pačią minutę, yra daugiau nei vienas, duomenų įterpimo algoritmas yra tas pats. Tik papildomai į analizę yra įtraukiamos ir naujos nuotraukos.

2.2. PRAŠOMŲ-SIŪLOMŲ KAINŲ SKIRTUMŲ ANALIZĖ[9]

1. Paprastas kainų skirtumas (*inside bid-ask spread*)

Pasirinktą laikotarpį daliname į vienos dienos trukmės laiko intervalus (dienos, kada buvo sustabdyta akcijos prekyba, į analizę neįtraukiamos). Kiekvieną dieną, nuo 10:00:00 iki 13:49:59, kas 10 s imame rinkos gylio lentelės nuotrauką. Iš jos, pagal (1.1) formulę, apskaičiuojame paprastą pirkimo-pardavimo kainų skirtumą. Tokiu būdu per dieną gauname 1374 skirtumus.

Kadangi duomenys buvo gauti tiesiogiai iš interneto kas tam tikrą laiko intervalą, tai egzistuoja laiko momentai, kada neturime rinkos gylio lentelių nuotraukų. Tokiu atveju, skaičiuojant paprastą kainų skirtumą, skaičiavimams naudojame artimiausia (laiko atžvilgiu) prieš tai buvusią rinkos gylio lentelės nuotrauką. Jei tokios nuotraukos nėra (tokia situacija susidaro dienos pradžioje), imam po to artimiausiai esančią rinkos gylio lentelės nuotrauką.

Paprasto kainų skirtumo vidurkį apskaičiuojame kaip visų skirtumų (per pasirinktą laikotarpį) aritmetinį vidurkį.

2. Efektyvus kainų skirtumas (*effective spread*)

Efektyvus pirkimo-pardavimo kainų skirtumas kiekvienam sandoriui apskaičiuojamas pagal (1.2) formulę. Pasirinkto laikotarpio vidutinis efektyvus kainų skirtumas – tai visų, per pasirinktą laikotarpį apskaičiuotų, efektyvių kainų skirtumų aritmetinis vidurkis.

Jei pasirinktu laikotarpiu neįvyko nei vienas sandoris, tuomet efektyvus kainų skirtumas nėra apskaičiuojamas.

[9] Kiekvienos pasirinktos akcijos analizės laikotarpis yra nuo 2008 02 25 iki 2008 04 18. Šio laikotarpio kiekvieną prekybos dieną analizuojami prekybos sesijos laiku (nuo 10:00:00 iki 13:49:59) pateikiami duomenys. Akcijos subrinkos prekybos dienos struktūra aprašyta 1-ame priede.

3. Užfiksuotas kainų skirtumas

Užfiksuotas kainų skirtumas – tai skirtumas tarp, vienas paskui kitą įvykusių, sandorių kainų. Pasirinkto laikotarpio vidutinis užfiksuotas kainų skirtumas – tai visų, per pasirinktą laikotarpį apskaičiuotų, užfiksuotų kainų skirtumų aritmetinis vidurkis.

Jei per pasirinktą laikotarpį neįvyko nei vienas sandoris arba įvyko tik vienas sandoris, tuomet užfiksuotas kainų skirtumas nėra skaičiuojamas.

4. Roll matas

Roll matas pasirinktam laikotarpiui apskaičiuojamas pagal (1.3) formulę. Akcijos rinkos kainų skirtumai imami kaip skirtumai tarp, vienas paskui kitą įvykusių, sandorių kainų. Autokovariacija skaičiuojama pagal (2.1) formulę.

$$\operatorname{cov}(\Delta p_t, \Delta p_{t-1}) = \frac{1}{T-1}\sum_{t=2}^{T} \Delta p_t \Delta p_{t-1} - \overline{\Delta p_t}\,\overline{\Delta p_{t-1}}; \qquad \textbf{(2.1)}$$

$$\text{čia } \overline{\Delta p_t} = \frac{1}{T-1}\sum_{t=2}^{T} \Delta p_t \text{ ir } \overline{\Delta p_{t-1}} = \frac{1}{T-1}\sum_{t=1}^{T-1} \Delta p_t$$

Visų, geriausių prašomų-siūlomų kainų skirtumų, vidurkių rezultatai pateikiami 2.2 lentelėje.

2.2 lentelė

Vidutiniai prašomų-siūlomų kainų skirtumai (akcijos rinkos kaina – įvykusio sandorio kaina)

Akcija	Data (nuo)	Data (iki)	Paprastas	Efektyvus	Užfiksuotas	"Roll" įvertis
ANK1L	2008.02.25	2008.04.18	0,080300	0,068947	-0,003889	0,0889
APG1L	2008.02.25	2008.04.18	0,072371	0,074616	-0,001986	0,0308
CTS1L	2008.02.25	2008.04.18	0,103819	0,091889	-0,004248	0,0186
GRG1L	2008.02.25	2008.04.18	0,136206	0,099540	-0,002326	0,0548
IVL1L	2008.02.25	2008.04.18	0,122778	0,105758	-0,001725	0,0227
LDJ1L	2008.02.25	2008.04.18	0,059733	0,050292	-0,000619	0,0228
NDL1L	2008.02.25	2008.04.18	31,084728	23,151493	-1,340909	11,3876
SAN1L	2008.02.25	2008.04.18	0,585499	0,410800	-0,006142	0,1128
TEO1L	2008.02.25	2008.04.18	0,010506	0,010726	-0,000009	0,0049

2.3. STACIONARUMO TYRIMAS

Akcijos sandorių kainų skirtumų stacionarumo tyrimas atliekamas 1.4.2 skyrelyje aprašytu būdu. Visas laikotarpis (2008 02 25 – 2008 04 18) padalinamas į smulkesnius, vienos dienos trukmės, laikotarpius. (Dienos, kai nebuvo įvykdytas nei vienas sandoris, į analizę neįtraukiamos.) Skaičiuojant kainų skirtumus dienos bėgyje, pirmas skirtumas buvo skaičiuojamas kaip pirmojo tos dienos įvykusio sandorio kainos skirtumas nuo praėjusios dienos paskutiniojo sandorio kainos. Likusieji skirtumai skaičiuojami įprastai. Toks pasirinkimas grindžiamas tuo, kad yra dienų, kai įvyksta tik vienas sandoris. Tačiau jų į analizę neįtrauki negalime.

Pavyzdžiui, turime tokią, ANK1L akcijos įvykusių sandorių, informaciją:

Data	Laikas	Kaina, Lt	Kiekis
2008.03.03	13:43	0,98	580
2008.03.04	-		
2008.03.05	11:02	0,99	1000
2008.03.06	12:26	0,98	420
2008.03.06	12:26	0,98	100
2008.03.06	13:26	0,95	1000

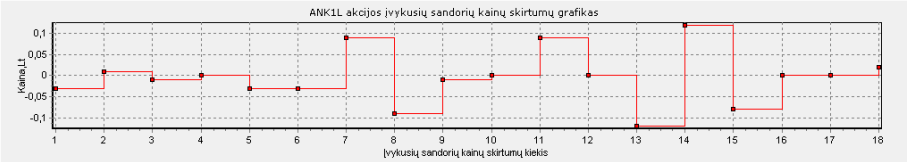
Tuomet stacionarumo analizei naudojami duomenys:

Data	Kainų skirtumas, Lt
2008.03.05	0,01
2008.03.06	-0,01
2008.03.06	0
2008.03.06	-0,03

Nepriklausomumui tarp aibės $x_1, x_2, \ldots, x_N$ elementų (mūsų atveju N - dienų skaičius) užtikrinimui, staciotarumo tyrime buvo imtas kas antras intervalas (kas antra diena).

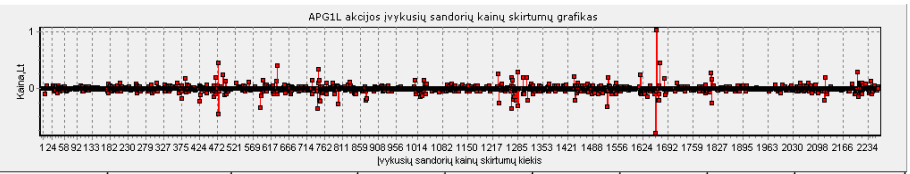
Pasirinktų akcijų įvykusių sandorių kainų skirtumų kitimo grafikai ir stacionarumo tyrimo rezultatai pateikti 2.7 paveikslėlyje.

ANK1L



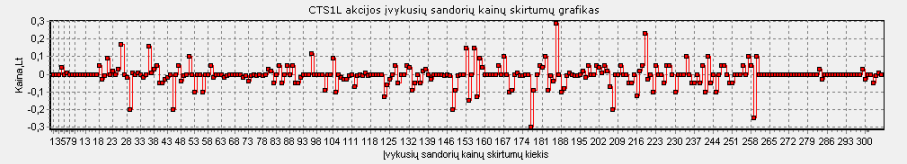
Akcija	Data (nuo)	Data (iki)		Alpha	A	Int_pr	Int_pb	
ANK1L	2008.02.25	2008.04.18	EX	0,01	6	0	13	---
ANK1L	2008.02.25	2008.04.18	EX	0,05	6	1	12	H0 priimti
ANK1L	2008.02.25	2008.04.18	E(X^2)	0,01	8	0	13	---
ANK1L	2008.02.25	2008.04.18	E(X^2)	0,05	8	1	12	H0 priimti

APG1L



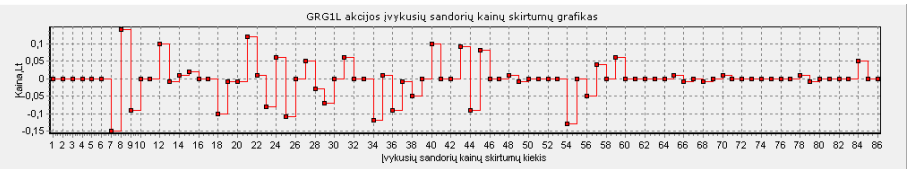
Akcija	Data (nuo)	Data (iki)		Alpha	A	Int_pr	Int_pb	
APG1L	2008.02.25	2008.04.18	EX	0,01	79	42	109	H0 priimti
APG1L	2008.02.25	2008.04.18	EX	0,05	79	50	101	H0 priimti
APG1L	2008.02.25	2008.04.18	E(X^2)	0,01	73	42	109	H0 priimti
APG1L	2008.02.25	2008.04.18	E(X^2)	0,05	73	50	101	H0 priimti

CTS1L



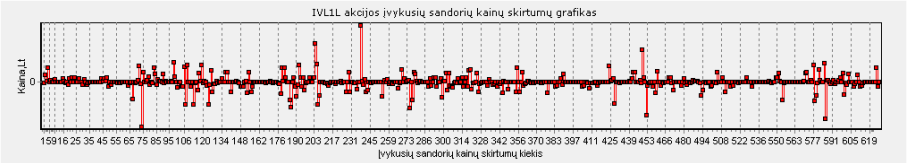
Akcija	Data (nuo)	Data (iki)		Alpha	A	Int_pr	Int_pb	
CTS1L	2008.02.25	2008.04.18	EX	0,01	74	36	99	H0 priimti
CTS1L	2008.02.25	2008.04.18	EX	0,05	74	44	91	H0 priimti
CTS1L	2008.02.25	2008.04.18	E(X^2)	0,01	76	36	99	H0 priimti
CTS1L	2008.02.25	2008.04.18	E(X^2)	0,05	76	44	91	H0 priimti

GRG1L



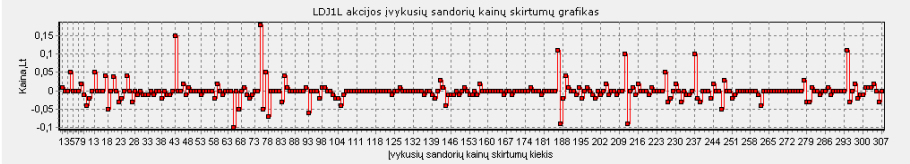
Akcija	Data (nuo)	Data (iki)		Alpha	A	Int_pr	Int_pb	
GRG1L	2008.02.25	2008.04.18	EX	0,01	37	21	68	H0 priimti
GRG1L	2008.02.25	2008.04.18	EX	0,05	37	27	62	H0 priimti
GRG1L	2008.02.25	2008.04.18	E(X^2)	0,01	46	21	68	H0 priimti
GRG1L	2008.02.25	2008.04.18	E(X^2)	0,05	46	27	62	H0 priimti

IVL1L



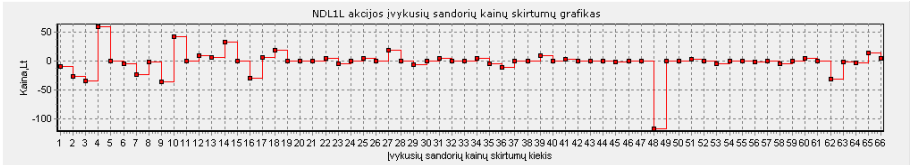
Akcija	Data (nuo)	Data (iki)		Alpha	A	Int_pr	Int_pb	
IVL1L	2008.02.25	2008.04.18	EX	0,01	90	42	109	H0 priimti
IVL1L	2008.02.25	2008.04.18	EX	0,05	90	50	101	H0 priimti
IVL1L	2008.02.25	2008.04.18	E(X^2)	0,01	77	42	109	H0 priimti
IVL1L	2008.02.25	2008.04.18	E(X^2)	0,05	77	50	101	H0 priimti

LDJ1L



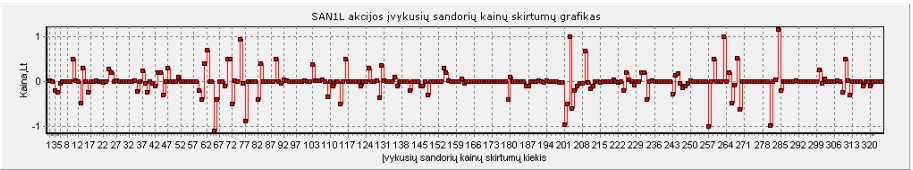
Akcija	Data (nuo)	Data (iki)		Alpha	A	Int_pr	Int_pb	
LDJ1L	2008.02.25	2008.04.18	EX	0,01	46	31	88	H0 priimti
LDJ1L	2008.02.25	2008.04.18	EX	0,05	46	38	81	H0 priimti
LDJ1L	2008.02.25	2008.04.18	E(X^2)	0,01	62	31	88	H0 priimti
LDJ1L	2008.02.25	2008.04.18	E(X^2)	0,05	62	38	81	H0 priimti

NDL1L



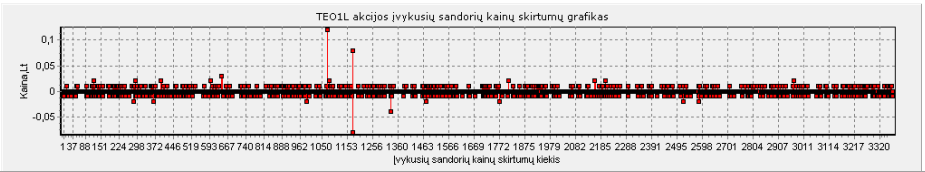
Akcija	Data (nuo)	Data (iki)		Alpha	A	Int_pr	Int_pb	
NDL1L	2008.02.25	2008.04.18	EX	0,01	10	1	18	H0 priimti
NDL1L	2008.02.25	2008.04.18	EX	0,05	10	3	16	H0 priimti
NDL1L	2008.02.25	2008.04.18	E(X^2)	0,01	9	1	18	H0 priimti
NDL1L	2008.02.25	2008.04.18	E(X^2)	0,05	9	3	16	H0 priimti

SAN1L



Akcija	Data (nuo)	Data (iki)		Alpha	A	Int_pr	Int_pb	
SAN1L	2008.02.25	2008.04.18	EX	0,01	78	42	109	H0 priimti
SAN1L	2008.02.25	2008.04.18	EX	0,05	78	50	101	H0 priimti
SAN1L	2008.02.25	2008.04.18	E(X^2)	0,01	64	42	109	H0 priimti
SAN1L	2008.02.25	2008.04.18	E(X^2)	0,05	64	50	101	H0 priimti

TEO1L



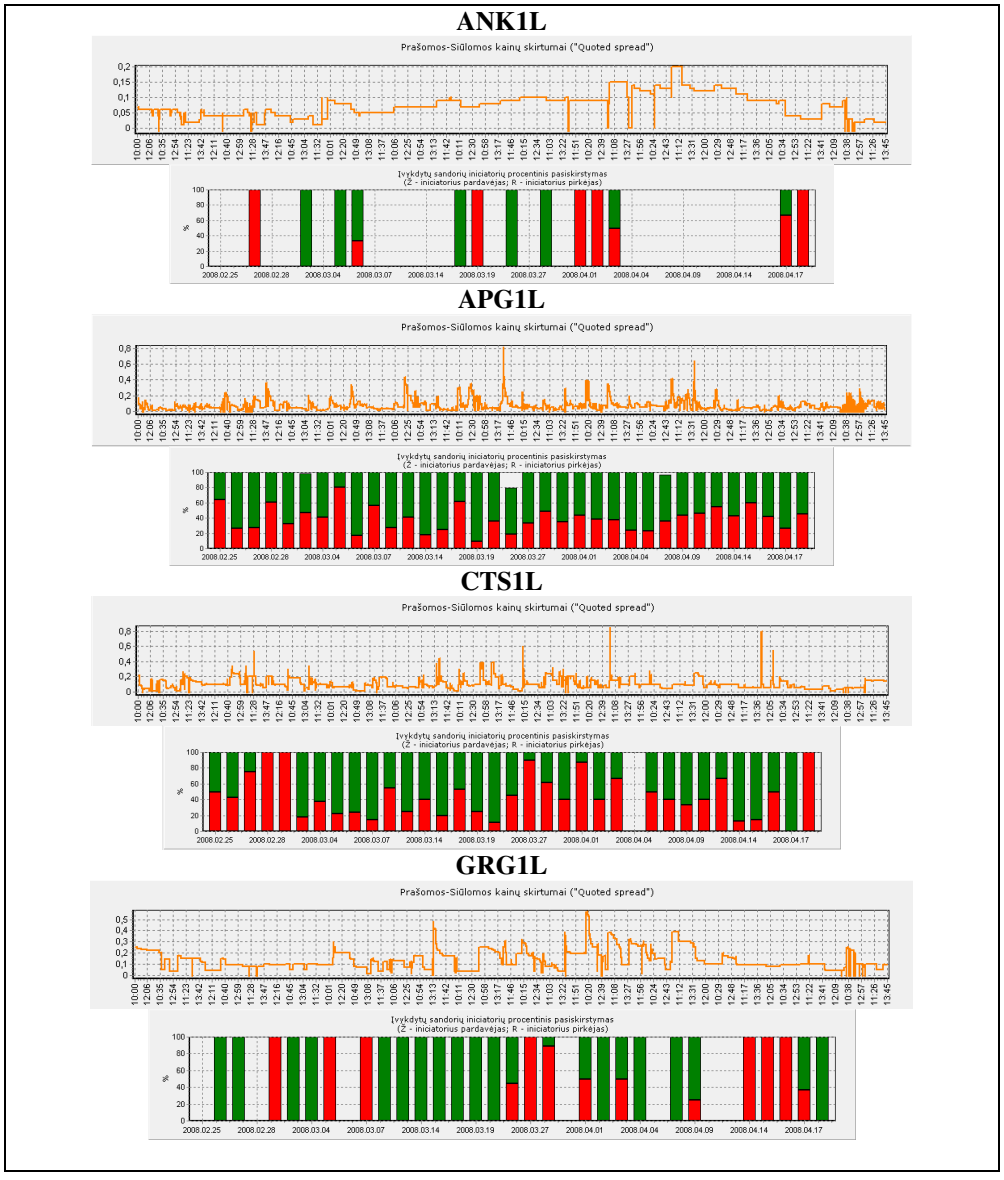
Akcija	Data (nuo)	Data (iki)		Alpha	A	Int_pr	Int_pb	
TEO1L	2008.02.25	2008.04.18	EX	0,01	77	42	109	H0 priimti
TEO1L	2008.02.25	2008.04.18	EX	0,05	77	50	101	H0 priimti
TEO1L	2008.02.25	2008.04.18	E(X^2)	0,01	84	42	109	H0 priimti
TEO1L	2008.02.25	2008.04.18	E(X^2)	0,05	84	50	101	H0 priimti

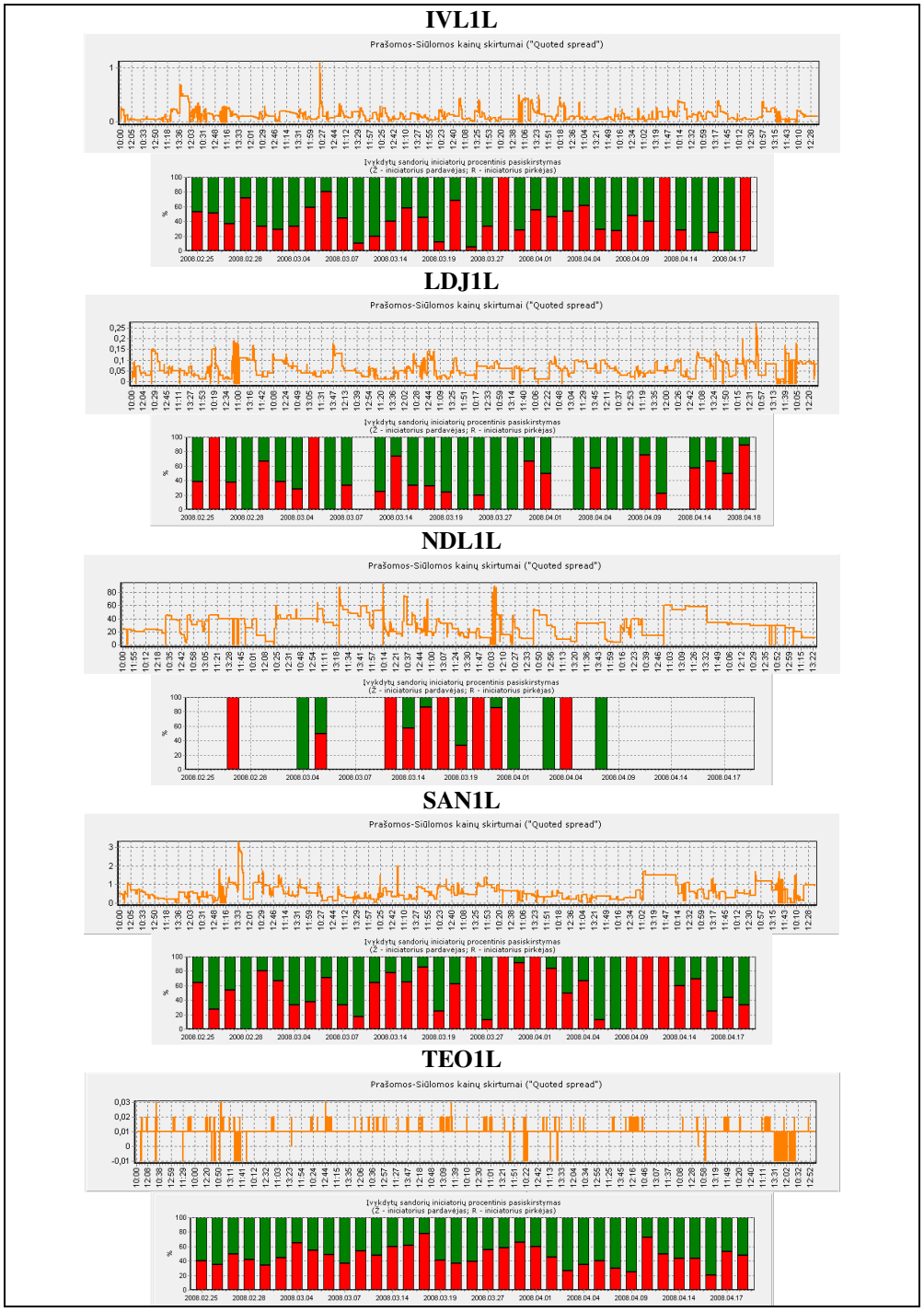
2.7 pav. Pasirinktų akcijų įvykusių sandorių kainų skirtumų stacionarumo tyrimo rezultatai

2.4. RINKOS EFEKTYVUMO TYRIMAS

Geriausios prašomos-siūlomos kainos skirtumo kitimo grafikas: skaičiuojami ir vaizduojami pasirinktos akcijos nurodyto laikotarpio kiekvienos minutės paprasto kainų skirtumo vidurkiai. Kiekvieną minutę išrenkamos skirtingos rinkos gylio lentelės nuotraukos ir iš kiekvienos jų skaičiuojamas paprastas kainų skirtumas. Grafike vaizduojamas kiekvieną minutę apskaičiuotų tokių skirtumų aritmetinis vidurkis. Tuo atveju, kai per minutę nėra nei vienos rinkos gylio lentelės nuotraukos, grafike vaizduojama neigiama vidurkio reikšmė (-0.01). Grafike paprasto kainų skirtumo vidurkio reikšmė 0 reiškia, kad tuo momentu rinkoje dalyvavo tik viena dalyvių pusė (pirkėjai arba pardavėjai).

Procentinis sandorių iniciatorių pasiskirstymas: pasirinktos akcijos kiekvieną, nurodyto laikotarpio, dieną skaičiuojama, kiek procentų visų įvykusių sandorių skaičiaus inicijavo pirkėjas, kiek pardavėjas.





2.8 pav. Pasirinktų akcijų geriausios prašomos-siūlomos kainos skirtumo kitimo ir įvykusių sandorių iniciatorių procentinio pasiskirstymo grafikai (2008 02 25 – 2008 04 18)

2.3 lentelė

Pasirinktų akcijų įvykusių sandorų informacijos suvestinė (2008 02 25 – 2008 04 18)

Akcija	Nuo	Iki	Dienų kiekis	Inic. pirkėjas, %	Inic. pardavėjas,%	Sand. kiekis	Sand. vertė, Lt
ANK1L	2008.02.25	2008.04.18	35	47,37	52,63	19	8296,01
APG1L	2008.02.25	2008.04.18	35	38,64	60,34	2262	7929443,50
CTS1L	2008.02.25	2008.04.18	35	39,41	60,59	307	1587822,30
GRG1L	2008.02.25	2008.04.18	35	33,33	66,67	87	215357,52
IVL1L	2008.02.25	2008.04.18	35	41,47	58,53	627	2470345,20
LDJ1L	2008.02.25	2008.04.18	34	41,88	58,12	308	1390182,80
NDL1L	2008.02.25	2008.04.18	32	62,69	37,31	67	954329,35
SAN1L	2008.02.25	2008.04.18	35	55,38	44,62	325	3182235,30
TEO1L	2008.02.25	2008.04.18	35	46,80	53,20	3374	27630043,00

3. PROGRAMINĖ REALIZACIJA IR INSTRUKCIJA VARTOTOJUI

Visi duomenys yra saugomi duomenų bazėje. Tam, kad programinė įranga prisijungtų prie šios DB, prieš pradedant dirbti reikia paleisti install_BDE.bat failą. Jis yra įrašytas pridėtame kompakte, kataloge ProgramineIranga//Install.

3.1.1. DUOMENŲ NUSKAITYMAS

Duomenų nuskaitymui iš VVPB puslapio [8] naudojama programa „DB_Paruosimas“. Duomenų bazės katalogas „DuomenuBaze“ (šio katalogo pavadinimo keisti negalima) ir duomenų nuskaitymo programinės įrangos katalogas „DBSkaitymas“ turi būti tame pačiame bendrame kataloge.

Programos pradinis langas pavaizduotas 3.1 pav. Žiūrėti ir valyti DB lentelėse esančius duomenis galima atsivertus langą „Duomenų bazė“ (3.2 pav.)

Failas→Uždaryti

Baigia programos darbą;

Duomenys→Nuskaitymas→Tiesioginių duomenų (3.3 pav.)

Skaito pasirinktų akcijų rinkos gylio lenteles nurodytu laiku. Duomenys saugomi „TRink_gylis.DBF“ faile (failo pavadinimo keisti negalima). Lentelės stulpelių pavadinimai ir eiliškumas turi būti (keisti negalima):

„DATA“ „TRUMPINYS“ „LAIKAS“ „KOMP_LAIK“ „PIRK_KIEK“ „PIRK_KAINA“ „PIRK_VAL“ „PARD_KAINA“ „PARD_VAL“ „PARD_KIEK“;

Duomenys→Nuskaitymas→Istorinių duomenų (3.4 pav.)

Skaito pasirinktų akcijų istorinius duomenis (įvykusių sandorų suvestines ir bendrą akcijų lentelę). Duomenys saugomi failuose „TSandoriai.dbf“ ir „TPagrindine.dbf“ (failų pavadinimų keisti negalima).

TSandoriai.dbf lentelės stulpelių pavadinimai ir eiliškumas (keisti negalima):

„DATA“ „TRUMPINYS“ „LAIKAS“ „KAINA“ „VALIUTA“ „KIEKIS“ „SAND_TIPAS“

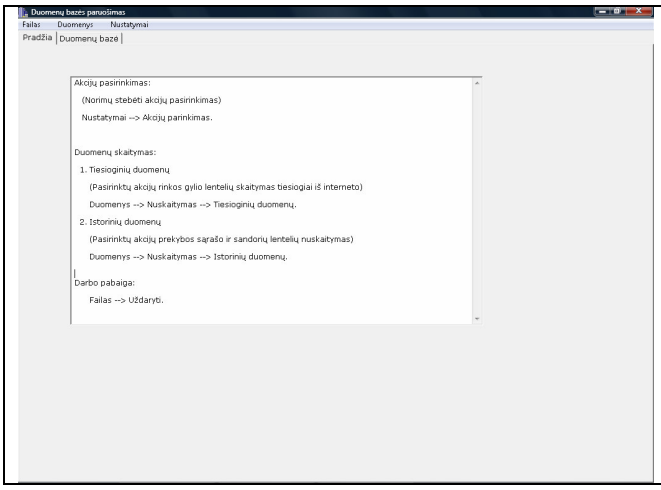
TPagrindine.dbf lentelės stulpelių pavadinimai ir eiliškumas (keisti negalima):

„DATA“ „TRUMPINYS“ „POK_PROC“ „PASK“ „VID“ „VAL“ „BIR“ „ATD“ „MAX_KAINA“ „MIN_KAINA“ „UZD“ „PERKA“ „PARD“ „SAND“ „KIEKIS“ „APYVARTA“ „LP“ **[A]**

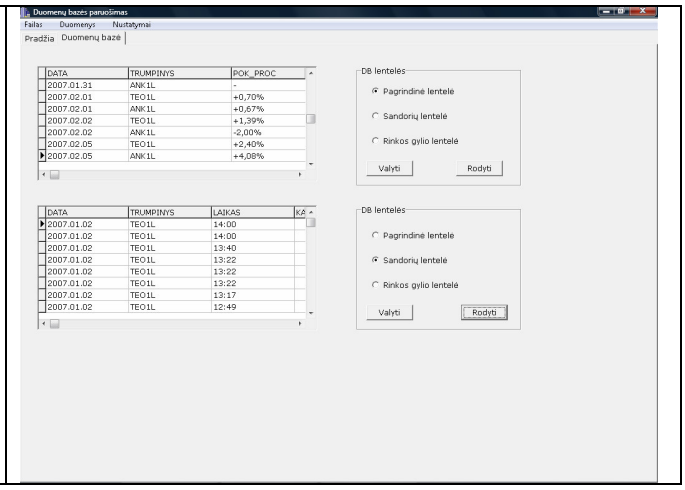
Nustatymai→Akcijų parinkimas (3.5 pav.)

Iš pateikto VVPB listinguojamų akcijų sąrašo pasirinkti, kurių akcijų duomenis skaityti. Jei nebus pasirinkta nei viena akcija, programa skaitys visų akcijų duomenis.

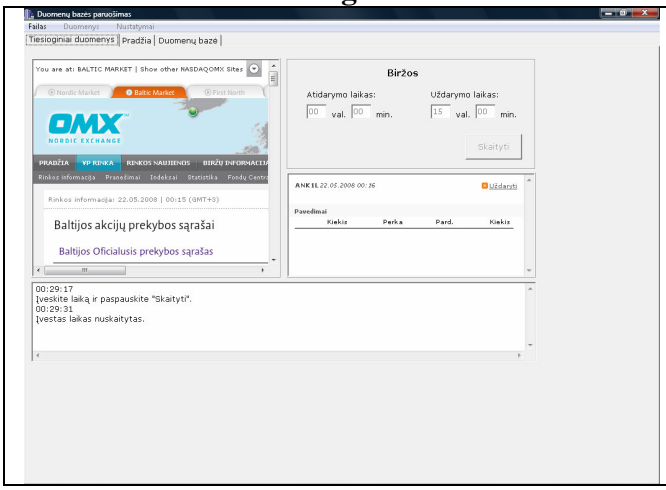
Programa „DB_Paruosimas“ yra jautri VVPB puslapio struktūros pakeitimams ir Windows Internet Explorer programos atnaujinimams (naujesnėse versijose reikia pažymėti: Tools→Internet Options→General→Settings(„Browsing history“)→Check for newer versions of stored pages→Every time I visit the webpage). Atsiradus tokiems pakitimams, programa gali netiksliai, arba visai duomenų nenuskaityti.



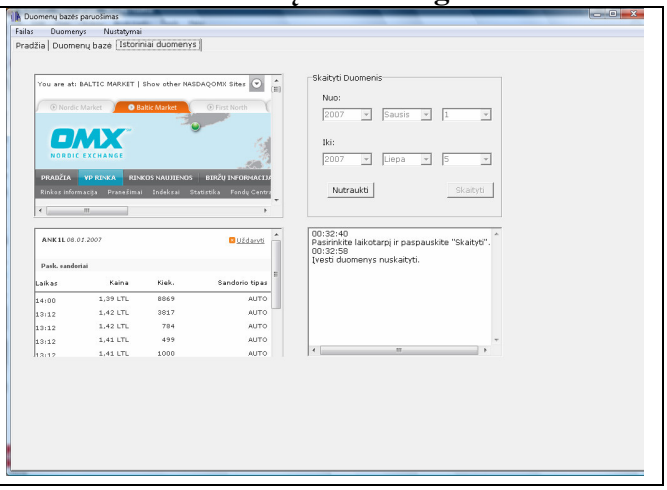
3.1 pav. Programos „DB_Paruosimas“ pradinis langas



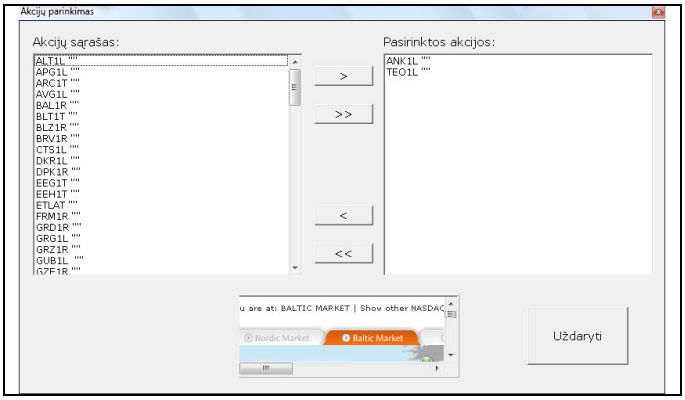
3.2 pav. Programos „DB_Paruosimas“ duomenų bazės langas



3.3 pav. Programos „DB_Paruosimas“ rinkos gylio lentelių skaitymo langas



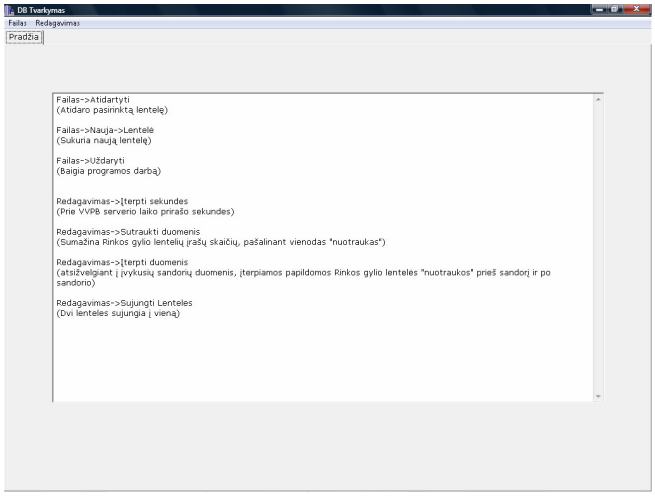
3.4 pav. Programos „DB_Paruosimas“ bendrosios ir įvykusių sandorių lentelių skaitymo langas



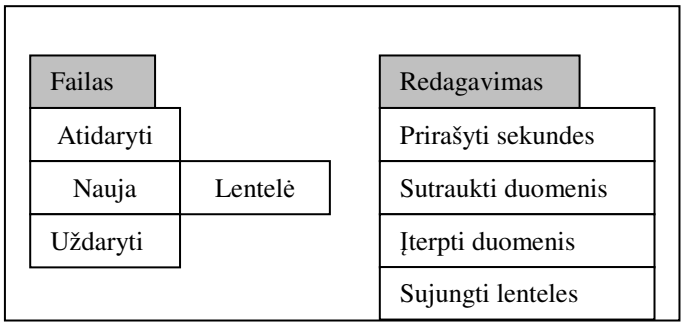
3.5 pav. Programos „DB_Paruosimas“ akcijų parinkimo langas

3.1.2. DUOMENŲ PARUOŠIMAS ANALIZEI

Programa „DB_Tvarkymas“ skirta duomenų bazės lentelių kūrimui, redagavimui, peržiūrai, taip pat duomenų paruošimui tolimesnei analizei.



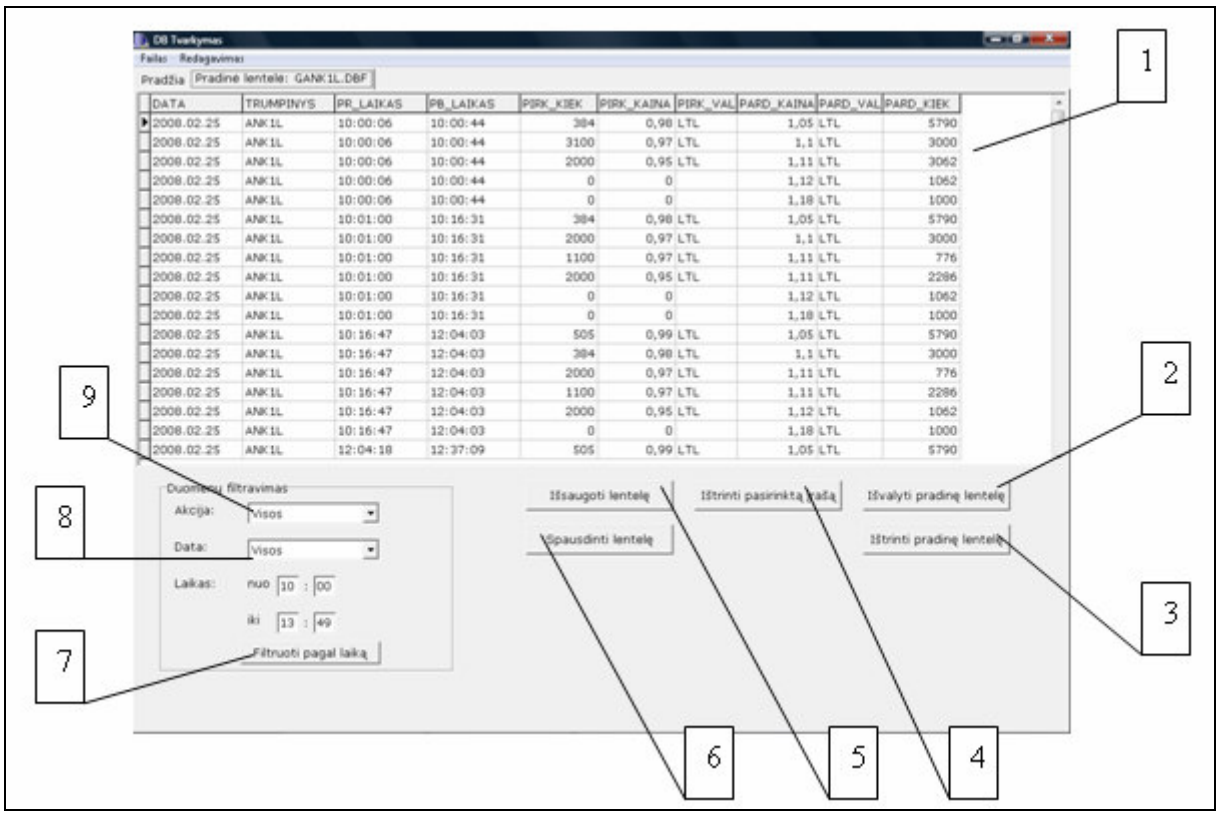
3.6 pav. Programos „DB_Tvarkymas“ pradinis langas



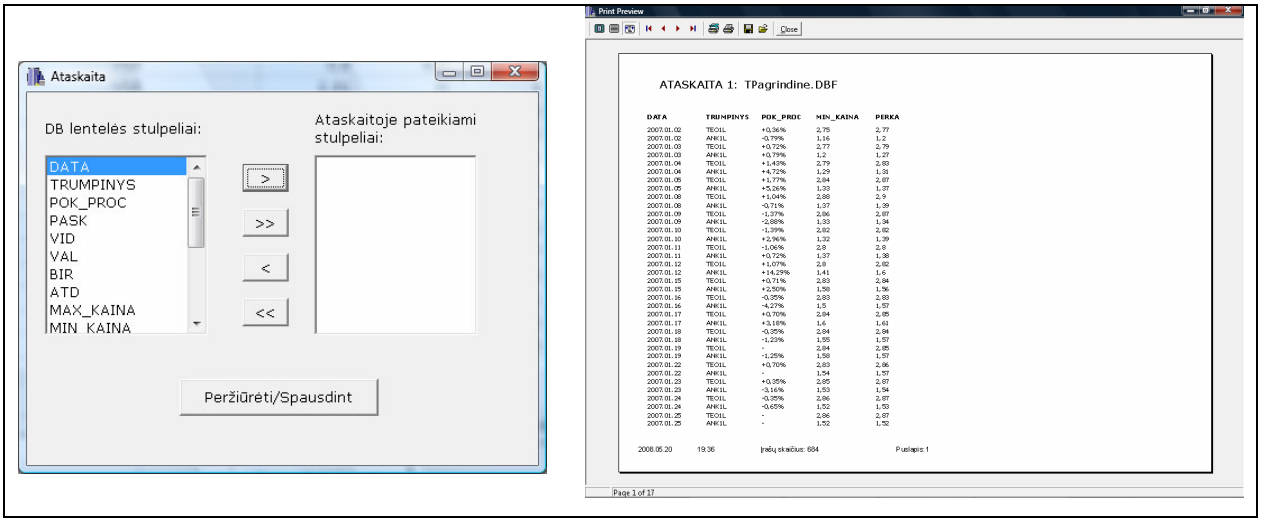
3.7 pav. Programos „DB_Tvarkymas“ meniu struktūra

Failas→Atidaryti (3.8 pav.)

1. Pasirinkta lentelė. Šiame lange lentelės duomenis galima redaguoti: pakeisti (du kartus spragtelint toje vietoje, kur norima keisti), arba įrašyti naujus (įrašymas galimas tik lentelės pabaigoje).
2. Ištrinami pasirinktos (atidarytos) lentelės duomenys.
3. Ištrinama pasirinkta (atidaryta) lentelė (t.y. ištrinamas lentelės failas).
4. Ištrinamas pasirinktas lentelės įrašas.
5. Lentelė, kuri matoma 1-ame lange, yra išsaugoma. Lentelės pavadinimas turi prasidėti raide, gali būti įtraukiami skaičiai ir „_“ simbolis.
6. Pasirinktos (atidarytos) lentelės peržiūra/spausdinimas (3.9 pav.)
7. Duomenų, pagal nurodytą laiką, filtravimas. Prieš filtruojant, reikia pažymėti stulpelį, pagal kurio duomenis bus atlikta filtracija.
8. Duomenų filtravimas pagal datą.
9. Duomenų filtravimas pagal akcijos pavadinimą.



3.8 pav. Vartotojo pasirinktos lentelės peržiūros/redagavimo langas



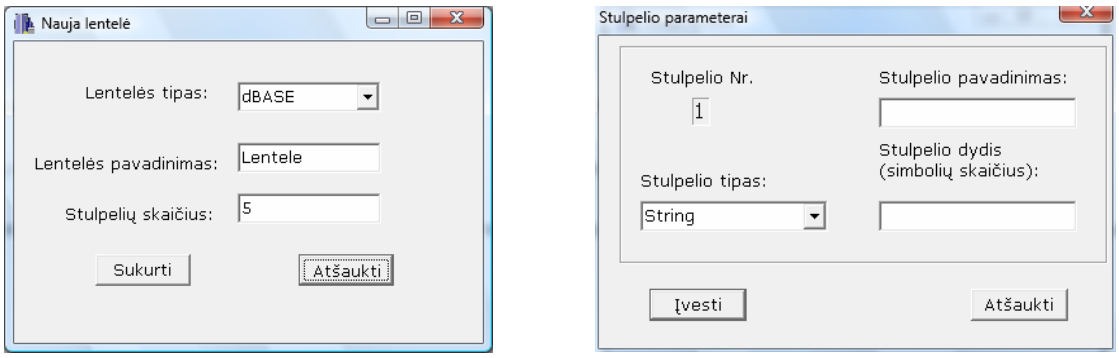
3.9 pav. Duomenų bazės lentelės peržiūros/spausdinimo langai

Failas→Nauja→Lentelė (3.10 pav.)

Sukuriama nauja lentelė. Lentelės ir stulpelių pavadinimai turi prasidėti raide, gali būti įtraukiami skaičiai ir „_“ simbolis.

Failas→Uždaryti

Baigia programos darbą.



3.10 pav. Naujos lentelės ir jos stulpelių parametrų įvedimo langai

Meniu komanda „Redagavimas“ yra skirta duomenų paruošimui tolimesnei analizei su programa „Duomenu_Analize“. Paruošime yra būtini pirmi trys žingsniai: „Prirašyti sekundes“, „Sutraukti duomenis“ ir „Įterpti duomenis“. (Jie turi būti atliekami išvardinta tvarka.)

Redagavimas→Prirašyti sekundes

Prie pasirinktos lentelės stulpelyje „LAIKAS“ esančių duomenų prirašo sekundes. Prieš tai programa reikalauja įvesti skaičių, kuris nurodo, į kokio dydžio intervalus dalinti dienos laiką (sekundžių įterpimo algoritmo pirmas žingsnis).

Pradinės lentelės stulpelių pavadinimai ir eiliškumas (keisti negalima):

„DATA“ „TRUMPINYS“ „LAIKAS“ (VVPB serverio laikas) „KOMP_LAIKAS“ (kompiuterio, kuriame paleista programa, laikas) „PIRK_KIEK“ „PIRK_KAINA“ „PIRK_VAL“ „PARD_KAINA“ „PARD_VAL“ „PARD_KIEK“

Rezultatai saugomi kitoje nurodytoje lentelėje. Jos struktūra turi būti tokia pati kaip ir pradinės lentelės.

Redagavimas→Sutraukti duomenis

Lygina rinkos gylio lentelės nuotraukas, einančias viena paskui kitą (laiko atžvilgiu). Sukuriami pradžios („Pr_laikas“) ir pabaigos („Pb_laikas“) laikai. Jie parodo, nuo kada iki kada rinkos gylio lentelėje nebuvo jokių pokyčių.

Pradinės lentelės stulpelių struktūra ir eiliškumas (keisti negalima):

„DATA“ „TRUMPINYS“ „LAIKAS“ (VVPB serverio laikas) „KOMP_LAIKAS“ (kompiuterio, kuriame paleista programa, laikas) „PIRK_KIEK“ „PIRK_KAINA“ „PIRK_VAL“ „PARD_KAINA“ „PARD_VAL“ „PARD_KIEK“

Saugojimo lentelės stulpelių struktūra ir eiliškumas (keisti negalima):

„DATA“ „TRUMPINYS“ „PR_LAIKAS“ „PB_LAIKAS“ „PIRK_KIEK“ „PIRK_KAINA“ „PIRK_VAL“ „PARD_KAINA“ „PARD_VAL“ „PARD_KIEK“

Redagavimas→Įterpti duomenis

Pagal duomenų įterpimo algoritmą įterpia papildomas rinkos gylio lentelės nuotraukas.

Pradinės ir naujos rinkos gylio lentelės stulpelių struktūra ir eiliškumas (keisti negalima):

„DATA“ „TRUMPINYS“ „PR_LAIKAS“ „PB_LAIKAS“ „PIRK_KIEK“ „PIRK_KAINA“ „PIRK_VAL“ „PARD_KAINA“ „PARD_VAL“ „PARD_KIEK“ **[B]**

Sandorių lentelės stulpelių struktūra ir eiliškumas:

„DATA“ „TRUMPINYS“ „LAIKAS“ „KAINA“ „VALIUTA“ „KIEKIS“ „SAND_TIPAS“

Programos kataloge „DBTvarkymas“ yra katalogas „Sandoriai“. Jame esančioje lentelėje „SandoriaiS.dbf“ yra saugomi modifikuoti sandorių duomenys:

„DATA“ „TRUMPINYS“ „LAIKAS“ „LAIKASS“„KAINA“ „VALIUTA“ „KIEKIS“ „SAND_TIPAS“ **[C]**

Šioje lentelėje įvedamas papildomas stulpelis „LAIKASS“. Jame rašomas rinkos gylio lentelės nuotraukos, kuri buvo prieš įvykdant šį sandorį, pabaigos laikas.

Redaguoti→Sujungti lenteles

Sujungia pasirinktas vienodos struktūros lenteles į vieną. Gauta lentelė įrašomą į pirmąją iš pasirinktų lentelių.

3.1.3. DUOMENŲ ANALIZĖ

Duomenų analizei sukurta programa „Duomenu_Analize“ yra kataloge DBAnalize. Visi duomenys, kuriuos naudoja ši programa, saugomi kataloge „Duomenys“ (DBAnalize//Duomenys). Duomenų katalogo vietos ir pavadinimo keisti negalima. Akcijų rinkos gylio lentelės saugomos „RGylis“ kataloge (DBAnalize//Duomenys//RGylis), sandorių – „Sandoriai“ (DBAnalize//Duomenys//Sandoriai), o bendroji lentelė – kataloge „Duomenys“ (DBAnalize//Duomenys). Kiekvienos akcijos rinkos gylio ir sandorių lentelės turi būti saugomos atskiruose failuose. Jų pavadinimai sudaromi tokiu būdu: „G“ + tikslus akcijos trumpinys, pvz. GANK1L.dbf, GTEO1L.dbf (rinkos gylio lentelei, struktūra [B]) ir „S“ + tikslus akcijos trumpinys, pvz. SANK1L, STEO1L, (sandorių lentelei, struktūra [C]). Bendrosios lentelės pavadinimas „TPagrindinė.dbf“ (struktūra [A]).

Failas→Uždaryti

Baigia programos darbą.

Analizė (3.11 pav.)

1. Prieš analizuojant duomenis, reikia užpildyti parametrų laukus: pasirinkti, kurios akcijos („Akcija“) ir nuo kada („nuo“) iki kada („iki“) atlikti analizę. Pasirinkus akciją, lange išvedama informacija apie jos turimus duomenis rinkos gylio, sandorių ir bendroje lentelėse. Datų laukai užpildomi atsižvelgiant į pasirinktos akcijos datas, esančias „TPagrindine.dbf“ lentelėje.
2. „Stacionarumo tyrimas“ (rezultatų langas 3.12 pav.)
   Tiriamas pasirinktos akcijos įvykusių sandorių kainų skirtumų stacionarumas (algoritmas aprašytas skyrelyje „Stacionarumo tyrimas“). „Imti kas antrą intervalą“ atžymėti tuo atveju, kai stacionarumo tyrimui imti ne kas antra, o kiekvieną dieną.
3. „Rinkos efektyvumo tyrimas“ (rezultatų langas 3.13 pav.)

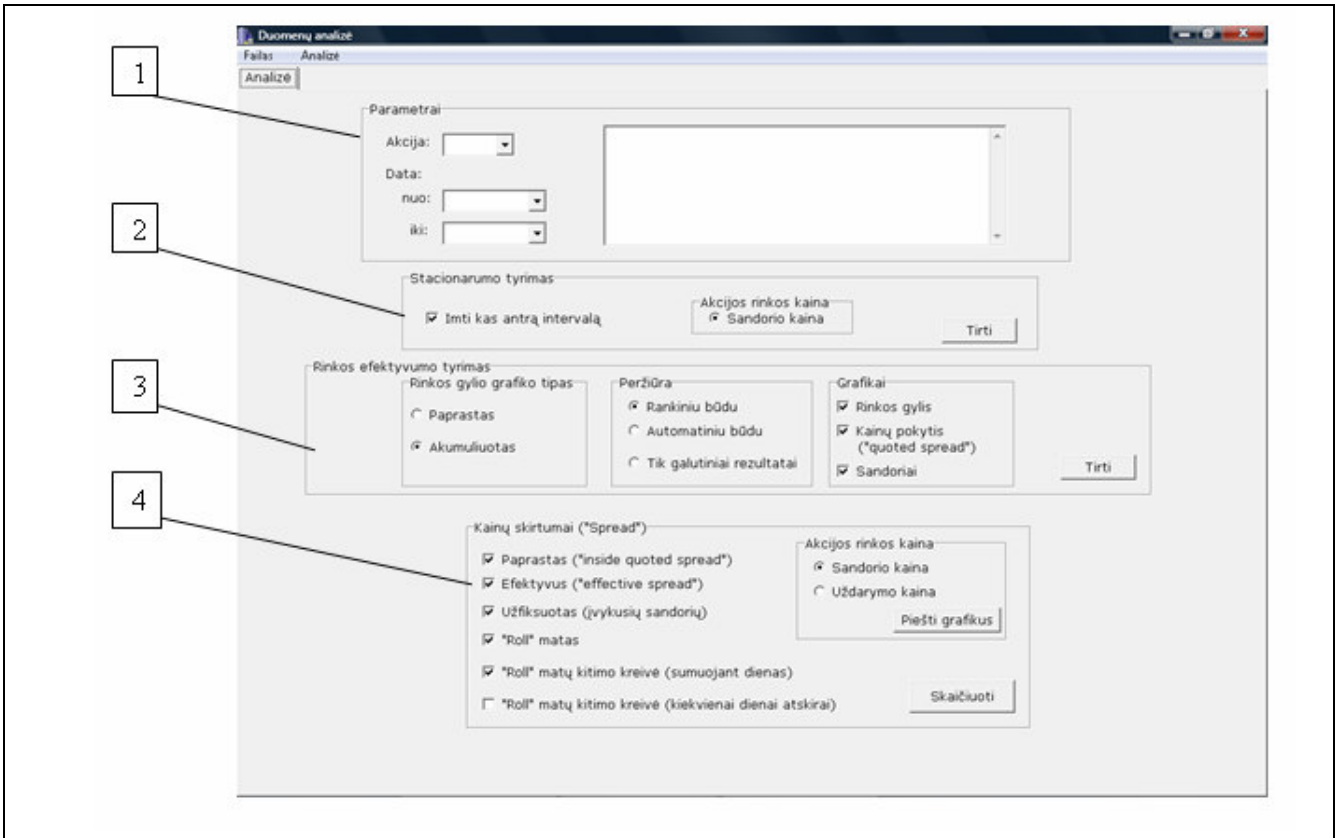
Rinkos gylio lentelės grafiko tipas – paprastas (grafike vaizduojami kiekvienos pavedimų kainos pirkti ir parduoti kiekiai) ir akumuliuotas (grafike vaizduojama akumuliuotas kiekio atžvilgiu grafikas; pvz. jei turime tokią pavedimų pirkti informaciją: 3,45 Lt 100 vnt. ir 3,44 Lt 200 vnt.; akumuliuotame grafike bus vaizduojama informacija: ties 3,45 – 100vnt., ties 3,44 – 100 + 200 = 300vnt.).

Peržiūra – rinkos efektyvumo tyrimo grafikų peržiūra tam tikrais laiko momentais. Rankiniu būdu – laiką keičiame rankiniu būdu, automatiniu būdu – pateikiami animuoti grafikai. Tik galutiniai rezultatai – išveda viso pasirinkto laikotarpio galutinius grafikus. Šiuo atveju, rinkos gylio lentelės grafikas nerodomas. Papildomai išvedama įvykusių sandorių informacija (procentinis sandorių iniciatorių pasiskirstymas).

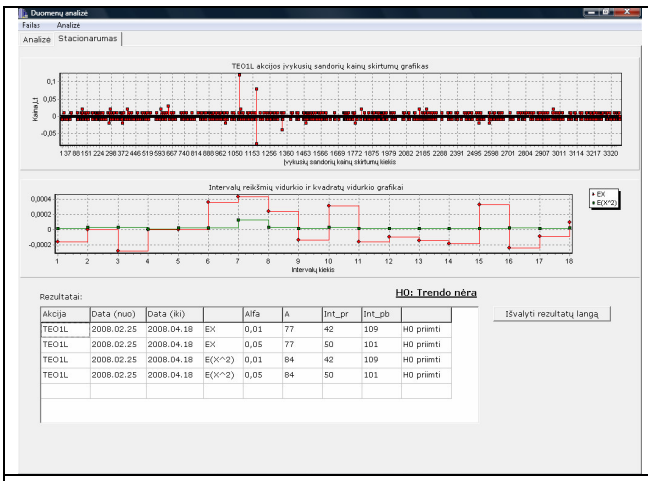
Grafikai - pažymime, kurią informaciją vaizduoti grafikuose.

4. „Kainų skirtumai“ (rezultatų langas 3.14 pav.)

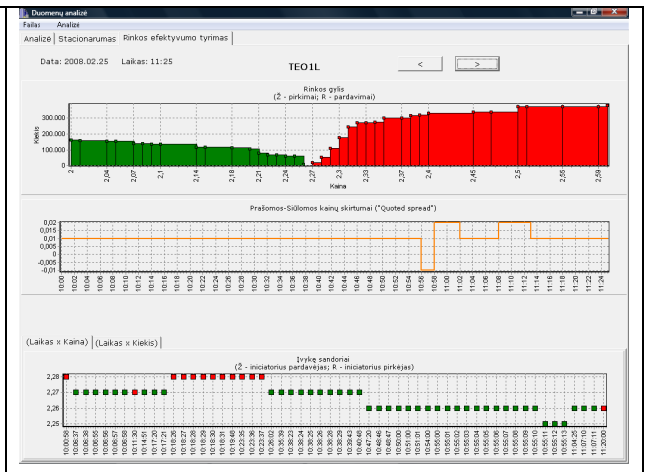
   Pažymėti, kuriuos kainų skirtumus, pasirinktai akcijai ir laikotarpiui, skaičiuoti. Roll matą skaičiuoja pagal tas akcijos rinkos kainas (įvykusių sandorių ar uždarymo), kurios pasirinktos skiltyje „Akcijos rinkos kaina“. Taip pat, paspaudus mygtuką „Piešti grafikus“, galima nubraižyti šių kainų ir jų skirtumų kitimo grafikus (rezultatai **3.15** pav.).



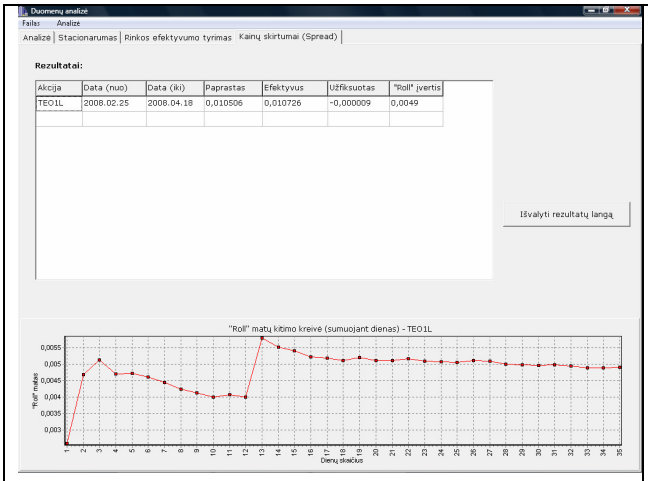
3.11 pav. Programos „Duomenu_Analize“ duomenų analizės langas



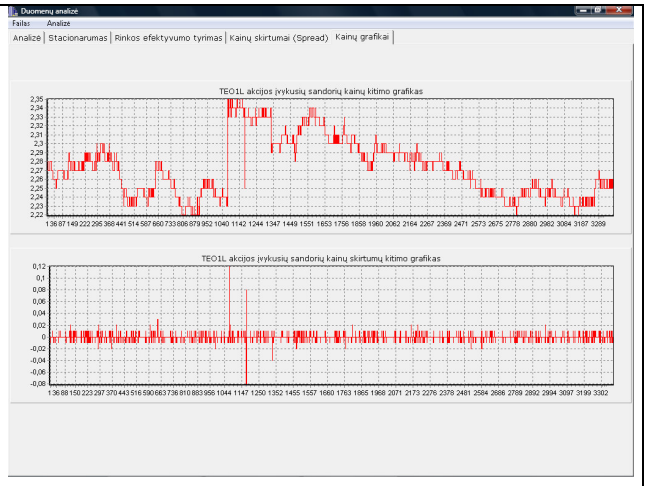
3.12 pav. Sandorių kainų skirtumų stacionarumo tyrimo rezultatų langas



3.13 pav. Rinkos efektyvumo tyrimo rezultatų langas



3.14 pav. Apskaičiuotų kainų skirtumų rezultatų langas



3.15 pav. Akcijos rinkos kainų ir jų skirtumų kitimo grafikai

4. DISKUSIJA

Atsižvelgiant į įvykdytų sandorių kiekį ir jų vertę (**2.3** lentelė), pasirinktas akcijas galima suskirstyti į 3 grupes:

1. Nelikvidžios akcijos (ANK1L, GRG1L, NDL1L);
2. Pusiau likvidžios akcijos (CTS1L, IVL1L, LDJ1L, SAN1L);
3. Likvidžios akcijos (APG1L, TEO1L).

Intuityviai mąstant, nelikvidžių akcijų rinka turėtų būti neefektyvi, o likvidžių priešingai – efektyvi. Rinkos efektyvumo tyrime galima pasinaudoti Roll matu. Esant efektyviai rinkai, jis turėtų gana tiksliai įvertinti efektyvų prašomos-siūlomos kainos skirtumą. Skaičiuojant šį matą, kaip įprasta technologija, dažniausiai skaičiavimuose yra naudojama akcijos uždarymo kaina. Tokį pasirinkimą lemia tai, kad ši informacija yra lengviau prieinama ir kiekvieną prekybos dieną ji egzistuoja. Kiek kitokia situacija yra su įvykusių sandorių kainomis. Informaciją ne taip lengva gauti, apdoroti ir paruošti skaičiavimams. O rezultatus gali iškreipti ir tai, kad yra dienų, kuomet neįvyko nei vienas sandoris. Atsiranda problema, kaip elgtis tokiu atveju: ar tokių dienų neįtraukti į analizę, ar apibrėžti taisykles, informacijos spragoms užpildyti. Jei akcijos įvykusio sandorio kainą renkamės kaip rinkos kainą, tai per diena įvykus daug sandorių[10], efektyvų kainų skirtumą galima įvertinti ir vienai dienai. Jei akcijos rinkos kaina imama uždarymo kaina, tai šiuo atveju, efektyvų kainų skirtumą jau galima įvertinti tik ilgesniam laikotarpiui. Remiantis antra Roll mato savybe (1.3 skyrius), jį skaičiuojant ilgam laikotarpiui, akcijos rinkos kainos parinkimas rezultatams įtakos neturėtų turėti.

Kaip matome iš 2.2 lentelės, Roll matas kiekvienai akcijai, nepriklausomai nuo to, ar ji likvidi, ar ne, nagrinėjamu laikotarpiu yra labai grubus (akcijos rinkos kaina buvo imama įvykusio sandorio kaina). 2.7 pav. pateikti stacionarumo tyrimo rezultatai rodo, kad visoms pasirinktoms akcijoms (reikšmingumo lygmuo $\alpha = 0.01$ ir $\alpha = 0.05$), hipotezė apie įvykusių sandorių kainų skirtumų proceso stacionarumą, priimama. Vadinasi, Roll mato paklaidos atsiranda dėl rinkos neefektyvumo, kurį galima patvirtinti ir kitais gautais rezultatais.

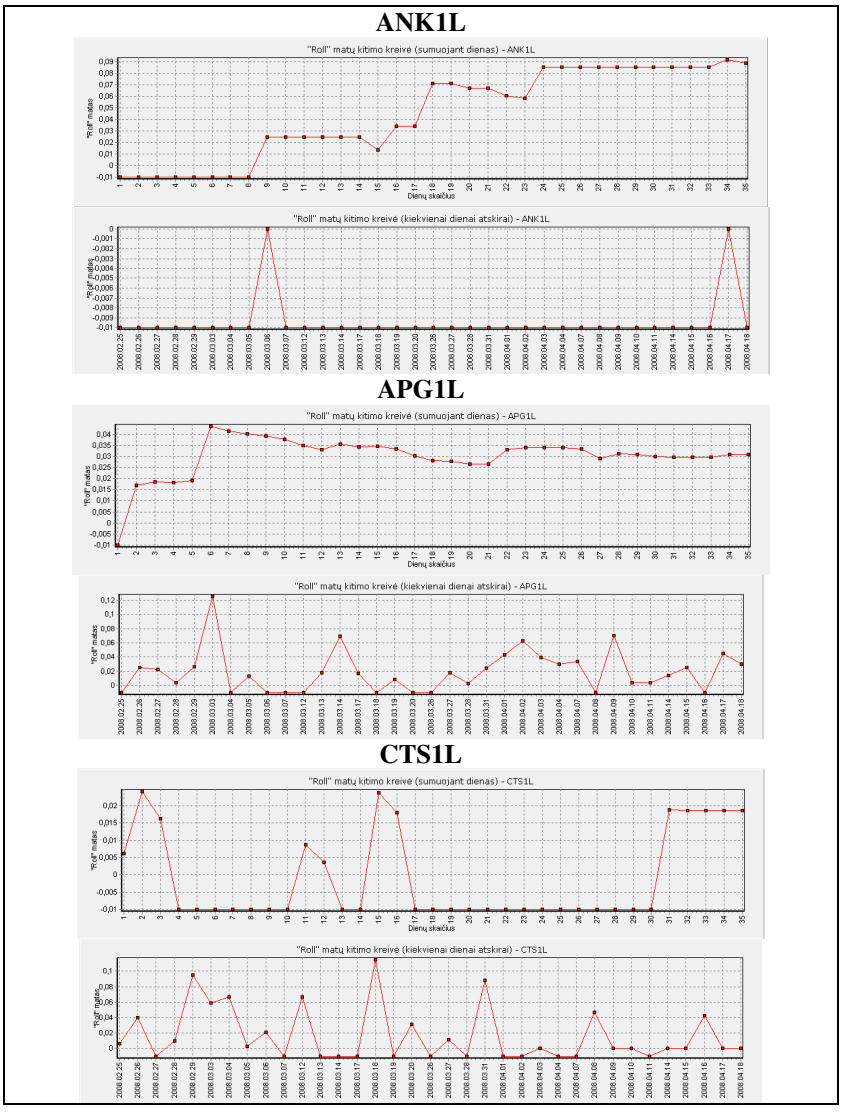
Vienas iš efektyvios rinkos požymių – geriausių prašomų-siūlomų kainų skirtumas išlieka vienodas. Iš **2.8** paveikslo matome, kad visoms akcijoms šis skirtumas svyruoja. Tik vienų daugiau, kitų mažiau. Apatinė skirtumo riba yra 0,01 Lt, kurią pasiekia tik TEO1L akcija.

Kiekvienos akcijos procentinis sandorių iniciatorių pasiskirstymo grafikas (**2.8** pav.) taip pat patvirtina rinkos neefektyvumą. Nelikvidžioms akcijoms šis grafikas visiškai neprognozuojamas – yra dienų, kada sandorių visai neįvyko, arba sandorius inicijuodavo tik viena rinkos dalyvių pusė (pirkėjai

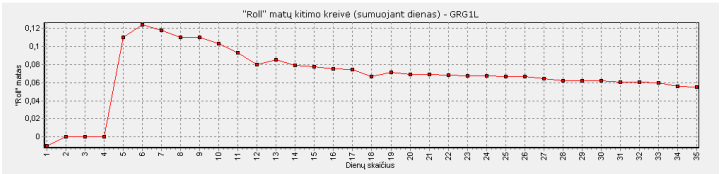
[10] Tikslaus skaičiaus, kiek sandorių turi įvykti, nėra. Galima orientuotis pagal tai, ar yra pakankamai laiko investuotojui įvertinti įvykusį sandorį. Pavyzdžiui, jei tarp sandorių yra 5 – 10 min. laiko tarpas, tai per valandą įvyksta nuo 6 iki 12, atitinkamai per prekybos sesiją – nuo 22 iki 47 sandorių. Šiuo atveju, investuotojas vertina pavienius sandorius, todėl galima sakyti, kad jų įvyko mažai. Kai investuotojas vertina įvykusių sandorių srautą, tada jau galima manyti, kad jų įvyko daug.

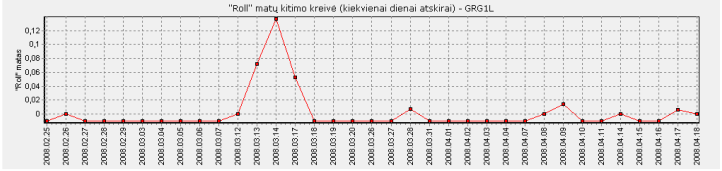
ar pardavėjai). Lyginant nelikvidžių ir pusiau likvidžių akcijų iniciatorių pasiskirstymo grafikus, galima pastebėti, kad pusiau likvidžių akcijų atveju, dienų skaičius, kada nėra nei vieno sandorio, pastebimai sumažėjo. Taip pat matoma, kad padidėjo dienų skaičius, kada per dieną sandorius inicijuodavo abi rinkos dalyvių pusės (pirkėjai ir pardavėjai). Likvidžių akcijų atveju, nagrinėjamu laikotarpiu neliko dienų, kada neįvyko nei vienas sandoris. Šių akcijų rinkos neefektyvumą parodo tai, kad sandorių iniciavime vis tik vyraudavo tai vieni, tai kiti.

4.1 pav. pavaizduoti pasirinktu laikotarpiu kiekvienai akcijai apskaičiuoti Roll matų kitimo grafikai (viršutinis grafikas – Roll matas skaičiuojamas suminiam dienų skaičiui, t.y. skaičiuojam matą pirmai pasirinkto laikotarpio dienai, paskui pirmoms dvejoms, pirmoms trejoms ir t.t.; apatinis – Roll matas skaičiuojamas kiekvienai dienai atskirai). Jei matas įgyja neigiama reikšmę (-0,01), tai reiškia, kad pagal (1.3) lygybę jo apskaičiuoti neįmanoma (priklausomai nuo situacijos, taip gali atsitikti, jei nėra įvykusių sandorių, arba per dieną įvyko tik vienas sandoris, taip pat jei įvykusių sandorių kainų skirtumai susieti teigiama kovariacija). Kaip matome iš pateiktų grafikų, nei vienai akcijai Roll matas nėra pastovus. Šis faktas vėlgi patvirtina rinkos neefektyvumą.

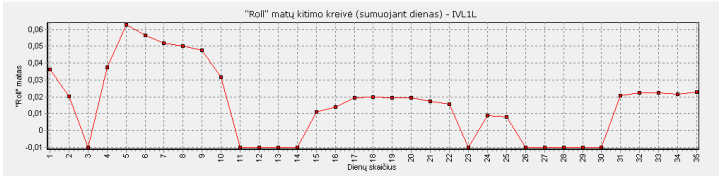


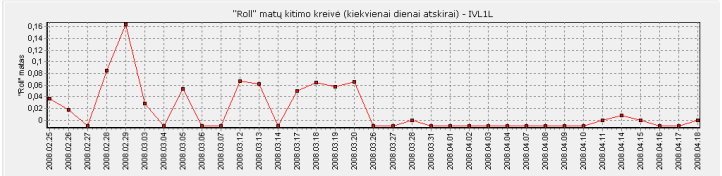
GRG1L



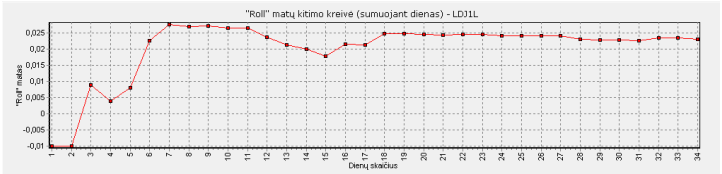


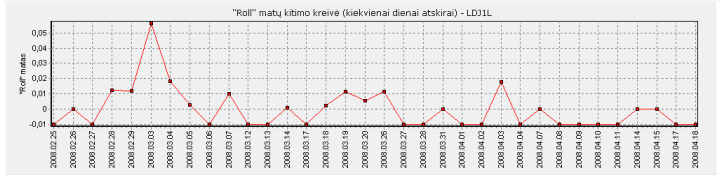
IVL1L



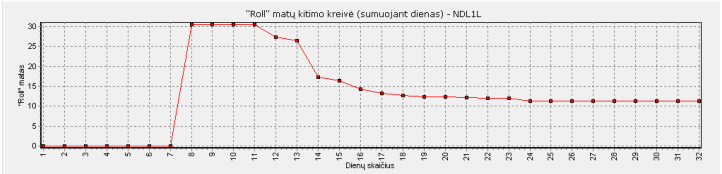


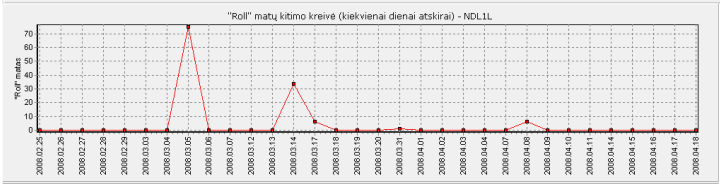
LDJ1L



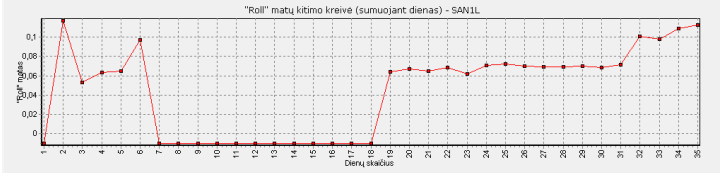


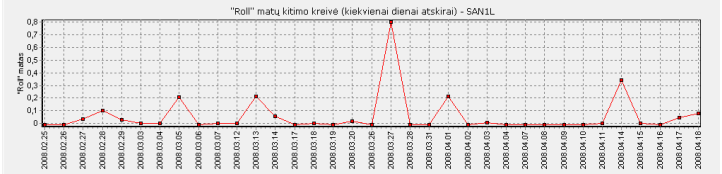
NDL1L

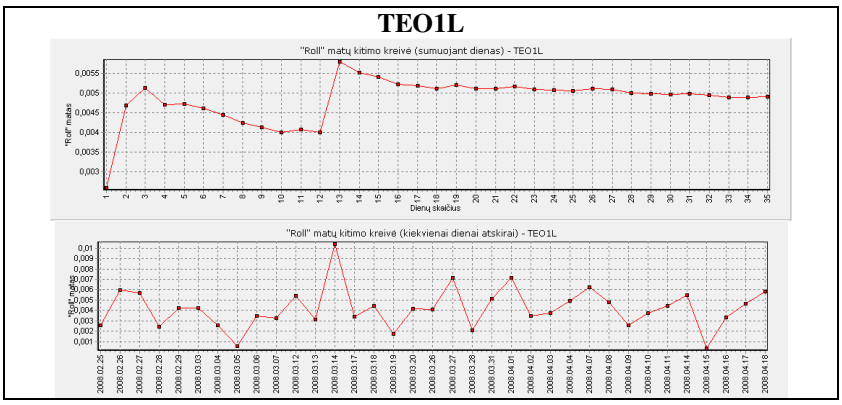




SAN1L



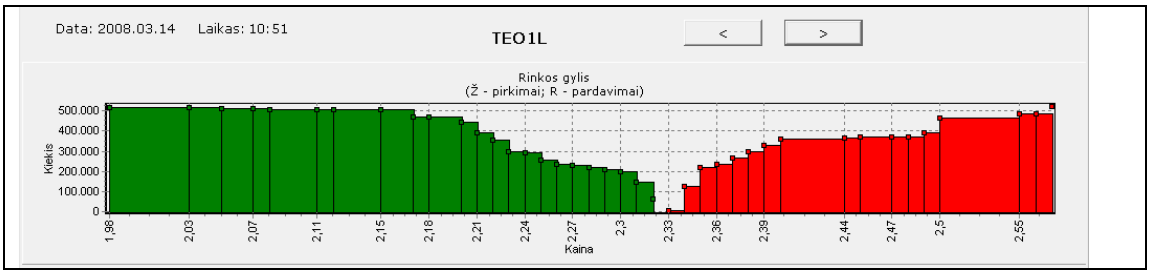




4.1 pav. Roll mato kitimo grafikai, kai akcijos rinkos kaina–įvykusio sandorio kaina (kiekvienos akcijos viršutinis grafikas – skaičiuojant suminiam dienų skaičiui, apatinis – skaičiuojant kiekvienai dienai atskirai)

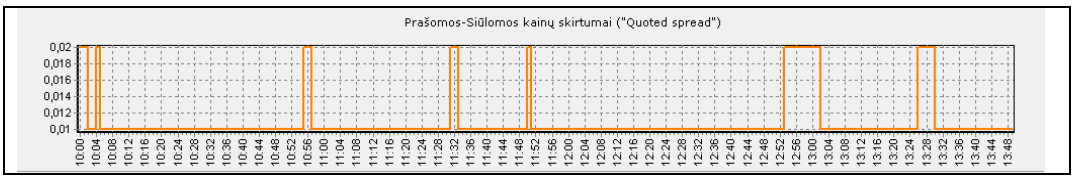
2 priede pateikiami geriausių prašomų-siūlomų kainų skirtumai ir Roll mato kitimo grafikai, kai akcijos rinkos kaina – akcijos uždarymo kaina. Kaip ir buvo galima tikėtis, dėl rinkos neefektyvumo Roll matas šiuo atveju taip pat yra labai grubus.

Panagrinėkime TEO1L akcijos geriausios pirkimo-pardavimo kainos skirtumus 2008.03.14 dieną. 4.2 pav. pavaizduotas rinkos gylio lentelės akumuliuotas grafikas (kiekvienai tolimesnei kainai kiekius sumuojame) laiko momentu 10:51. Kaip matome, tiek pavedimų pirkti, tiek parduoti, suminiai akcijų kiekiai yra apytiksliai lygūs. Stebint visos prekybos sesijos laikotarpio grafiko animaciją, galima pastebėti, kad ilgesnį laiką šie kiekiai išlieka apylygiai.



4.2 pav. Rinkos gylio lentelės akumuliuotas grafikas (TEO1L, 2008.03.14, 10:51)

4.3 pav. matome, kad geriausių prašomų-siūlomų kainų skirtumai praktiškai yra pastovūs. Iš visų 229 min., 21-os (9%) paprastas kainų skirtumas nukrypo nuo 0,01 Lt iki 0,02 Lt. Nepaisant to, kad daugiau sandorių inicijavo pirkėjai (4.1 lentelė), galima teigti, kad šią dieną rinkos efektyvumas dalinai susibalansuoja. Suskaičiuoti geriausių prašomų-siūlomų kainų skirtumų vidurkiai pateikti 4.2 lentelėje. Matome, kad šios dienos Roll matas nėra toks grubus (santykinės paklaidos nuo paprasto ir efektyvaus kainų skirtumų atitinkamai yra 0,058 ir 0,0925).



4.3 pav. Prašomų-siūlomų kainų skirtumų grafikas (TEO1L, 2008.03.14)

4.1 lentelė

Įvykusių sandorų informacijos suvestinė (TEO1L, 2008.03.14)

Akcija	Nuo	Iki	Dienų kiekis	Inic. pirkėjas, %	Inic. pardavėjas,%	Sand. kiekis	Sand. vertė, Lt
TEO1L	2008.03.14	2008.03.14	1	59,53	40,47	257	2565231,80

4.2 lentelė

Vidutiniai prašomų-siūlomų kainų skirtumai (akcijos rinkos kaina – įvykusio sandorio kaina, TEO1L, 2008.03.14)

Akcija	Data (nuo)	Data (iki)	Paprastas	Efektyvus	Užfiksuotas	"Roll" įvertis
TEO1L	2008.03.14	2008.03.14	0,011000	0,011362	-0,000039	0,0104

IŠVADOS

- Visų 9 akcijų įvykusių sandorių kainų skirtumai yra stacionarūs;
- Visų 9 akcijų rinka nėra efektyvi;
- Ir tuo atveju, kai akcijos rinkos kaina yra uždarymo kaina, ir kai sandorio kaina, Roll matas galėjo tik grubiai įvertini akcijos efektyvų kainų skirtumą nagrinėjamu laikotarpiu.
- Jei rinka efektyvi ir per dieną įvyksta daug sandorių, Roll matą galima naudoti vienos dienos efektyviam akcijos kainų pokyčiui apskaičiuoti.

LITERATŪRA

1. Bendat, J.S. and Persol, A.G. Random Data: Analysis and Measurement Procedures, 3rd Edition, John Wiley & Sons, NY, 2000.
2. Fielitz, B.D. Stationarity of random data: some implications for the distribution of stock price chantes, *The Journal of Financial and Quantitative Analysis*, Vol. 6, No. 3 (Jun., 1971), pp. 1025 – 1034.
3. Kancerevyčius, G. Finansai ir investicijos. Kaunas, 2006.
4. *NIST/SEMATECH e-Handbook of Statistical Methods*, http://www.itl.nist.gov/div898/handbook/
5. Roll, R. A simple implicit measure of the effective bid-ask spread in an efficient market, *The Journal of Finance*, Vol. 39, No. 4 (Sep., 1984), pp. 1127 – 1139.
6. Schultz, P. Regulatory and legal pressures and the cost of Nasdaq trading, *The Review of Financial Studies*, Vol. 13, No. 4 (2000), pp. 917 – 975.
7. Vo, M.T. Limit orders and the intraday behavior of market liquidity: Evidence from the Toronto stock exchange, *Global Finance Journal*, Vol. 17, No. 3 (March, 2007), pp. 379 – 396.
8. http://www.baltic.omxgroup.com/index.php?id=3040

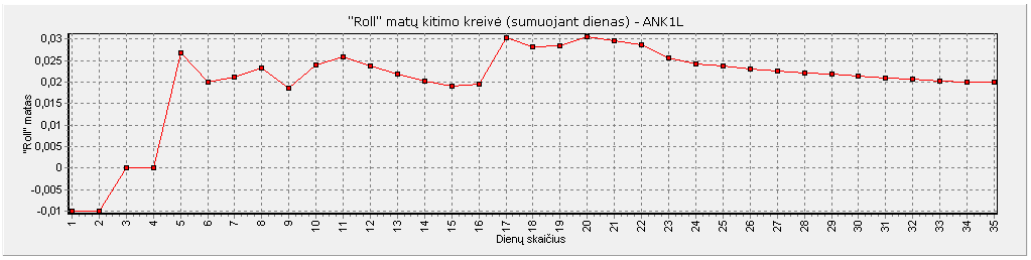
1 PRIEDAS
AKCIJŲ SUBRINKOS PREKYBOS DIENOS STRUKTŪRA

Laikas	Prekybos etapas
08:30 – 09:45	Priešprekybinė sesija
09:45 – 10:00	Laikas iki atidarymo aukciono
10:00	Atidarymo aukcionas
10:00 – 13:50	Prekybos sesija
13:50 – 14:00	Laikas iki uždarymo aukciono
14:00	Uždarymo aukcionas
14:05 – 14:30	Poprekybinė sesija

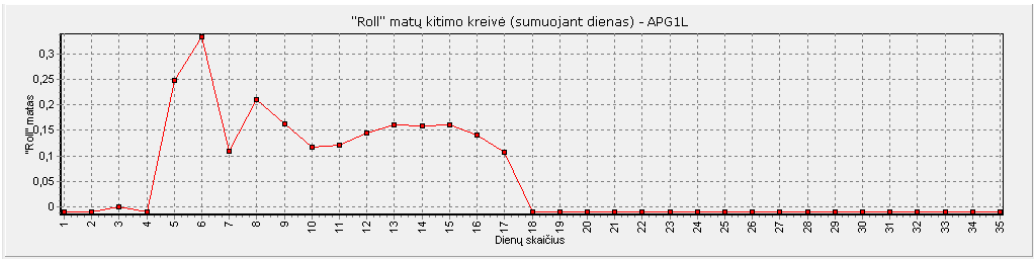
1. Priešprekybinė sesija (nuo 8:30 iki 9:45) – tai laikas, kuomet pavedimai pirkti ir parduoti yra įvedami, keičiami, anuliuojami, sustabdomas ar atnaujinamas jų vykdymas. Per šį prekybos etapą taip pat gali būti pranešama apie sudarytus tiesioginius sandorius. Prieš prekybinės sesijos metu makleriai gali matyti tik savo pavedimus.
2. Laikas iki atidarymo aukciono (nuo 09:45 iki 10:00) yra laikas, iki atidarymo aukciono, kai pavedimai teikiami, keičiami, atšaukiami, sustabdomas ar atnaujinamas jų vykdymas. Per laiką iki atidarymo aukciono makleriai gali matyti pavedimų knygoje apibendrintai pateiktas penkias geriausias kainas.
3. Per atidarymo aukcioną pavedimai įvykdomi aukciono kursu. Aukciono kursas nustatomas tik tuo atveju, jeigu konkrečioje pavedimų knygoje didžiausia pirkimo kaina yra didesnė arba lygi mažiausiai pardavimo kainai.
4. Prekybos sesija (nuo 10:00 iki 13:50) yra laikotarpis, per kurį pavedimai įvedami į prekybos sistemą ir įvykdomi sandoriai. Sandoriai gali būti įvykdomi dviem būdais:
   4.1. *Automatinio įvykdymo sandoriai* – pavedimai pirkti ir parduoti įvykdomi automatiškai prekybos sistemoje, laikantis kainos ir laiko prioriteto.
   4.2. *Tiesioginiai sandoriai*– tai sandoriai, kuriuos vertybinių popierių biržos nariai sudaro už sistemos ribų ir įveda į prekybos sistemą per tris minutes nuo sandorio sudarymo.
5. Laikas iki uždarymo aukciono (nuo 13:50 iki 14:00) yra laikas, per kurį pavedimai yra įvedami, keičiami, sustabdomi ar anuliuojami. Per šį laiką makleriai gali matyti pavedimų knygoje apibendrintai pateiktas penkias geriausias kainas.
6. Per uždarymo aukcioną pavedimai yra įvykdomi pavedimų knygoje aukciono kursu. Aukciono kursas nustatomas tik tuo atveju, jeigu konkrečioje pavedimų knygoje didžiausia pirkimo kaina yra didesnė arba lygi mažiausiai pardavimo kainai.
7. Poprekybinės sesijos (nuo 14:05 iki 14:30) metu galima tik pranešti apie sudarytus tiesioginius tiesioginiai sandorius. Makleriai negali įvesti ar keisti pavedimų, bet gali anuliuoti pavedimus.

2 PRIEDAS
ROLL MATO SKAIČIAVIMO REZULTATAI, KAI AKCIJOS RINKOS KAINA – UŽDARYMO KAINA

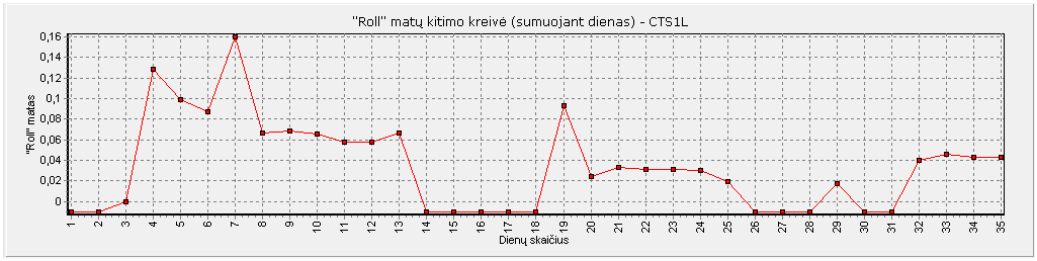
ANK1L



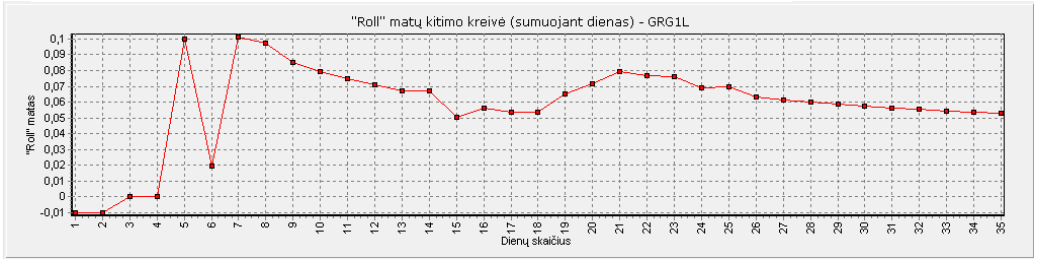
APG1L



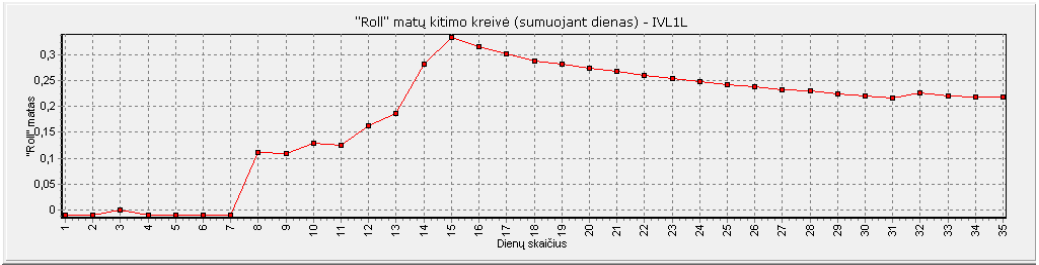
CTS1L



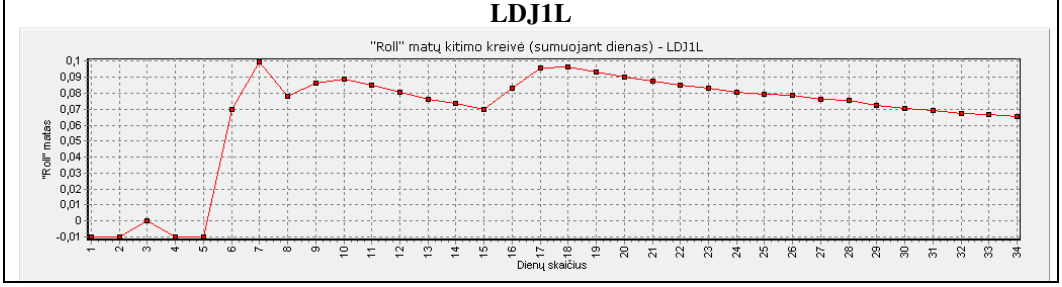
GRG1L

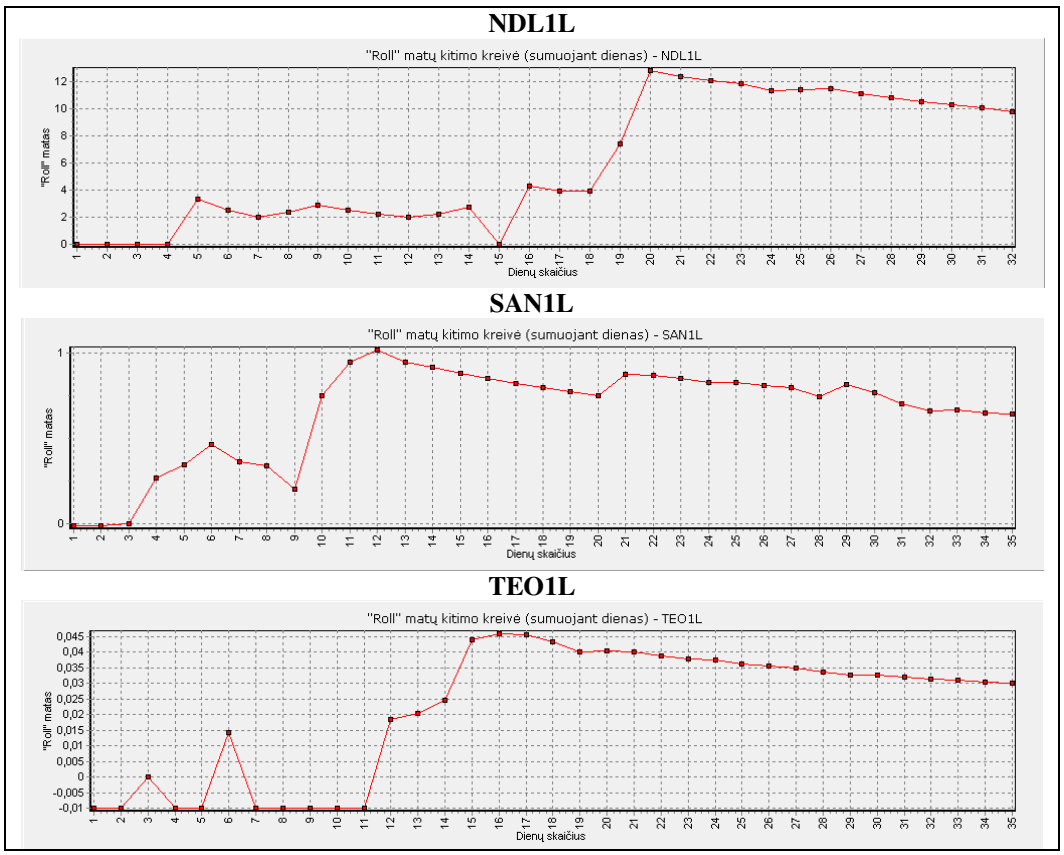


IVL1L



LDJ1L





Pasirinktų akcijų Roll mato kitimo grafikai (skaičiuojant suminiam dienų skaičiui), kai akcijos rinkos kaina – uždarymo kaina

Vidutiniai prašomų-siūlomų kainų skirtumai (akcijos rinkos kaina – uždarymo kaina)

Akcija	Data (nuo)	Data (iki)	Paprastas	Efektyvus	Užfiksuotas	"Roll" įvertis
ANK1L	2008.02.25	2008.04.18	0,080300	0,068947	-0,003889	0,0200
APG1L	2008.02.25	2008.04.18	0,072371	0,074616	-0,001986	---
CTS1L	2008.02.25	2008.04.18	0,103819	0,091889	-0,004248	0,0430
GRG1L	2008.02.25	2008.04.18	0,136206	0,099540	-0,002326	0,0527
IVL1L	2008.02.25	2008.04.18	0,122778	0,105758	-0,001725	0,2192
LDJ1L	2008.02.25	2008.04.18	0,059733	0,050292	-0,000619	0,0654
NDL1L	2008.02.25	2008.04.18	31,084728	23,151493	-1,340909	9,8083
SAN1L	2008.02.25	2008.04.18	0,585499	0,410800	-0,006142	0,6429
TEO1L	2008.02.25	2008.04.18	0,010506	0,010726	-0,000009	0,0300

3 PRIEDAS
PROGRAMŲ TEKSTŲ IŠTRAUKOS

„DB_Tvarkymas“

IterptiSekundes.h

```
//---------------------------------------------------------------------------
struct SLaikas
{
 SLaikas(AnsiString l1, AnsiString l2, AnsiString l3)
 {laikasS = l1; laikasK = l2; laikasP = l3;}
 AnsiString laikasS; //serverio laikas
 AnsiString laikasK; //kompiuterio laikas
 AnsiString laikasP;  //naujai suformuotas laikas
};
//---------------------------------------------------------------------------
class TIterptiSekundes{

   public:
      void
SekundziuIterpimas(TCheckListBox*,TCheckListBox*,TCheckListBox*,TQuery*,TLabel*,TLabel*,TTable*,TTable*
,TMemo*,TComboBox*); //prie minučių prirašo sekundes
      void
FormuotiLaikoIntervalus(TCheckListBox*,TCheckListBox*,TCheckListBox*,TQuery*,TTable*,TTable*,AnsiString
,AnsiString,int); //formuoja laiko intervalus sekundžių skaičiavimui
      AnsiString
RastiPirmaPaskutiniLaika(TQuery*,AnsiString,AnsiString,AnsiString,AnsiString&,AnsiString&); //randa tam
tikros akcijos ir tam tikros dienos pirmą laiką, nuo kurio pradedami formuoti laiko intervalai
      void FormuotiIntervalus(TCheckListBox*,AnsiString,AnsiString,int); //formuoja intervalus
      void
SkaiciuotiSekundes(TCheckListBox*,TQuery*,TTable*,TTable*,AnsiString,AnsiString,AnsiString,AnsiString,i
nt); //skaičiuoja kokios sekundės turi būti prirašytos
      void SudarytiLaikuVektoriu(TQuery*,AnsiString,AnsiString,AnsiString,AnsiString,vector
<SLaikas>&);   //sudaro reikiamų laikų vektorių
      void PabLaikasInt(AnsiString, AnsiString &,int); //randa intervalo pabaigos laiką
      void SekundesIntervale
(TTable*,TTable*,vector<SLaikas>&,vector<SLaikas>&,AnsiString,AnsiString,AnsiString,AnsiString);
//atitinkamame intervale randa sekundes
      int LaikasMinSek(AnsiString); //paverčia laiką minutėmis arba sekundėmis
      void RastiIndeksus(vector <SLaikas>,int,int&,int&); //randa lango pradžios ir pabaigos
indeksus
      void ApskaiciuotiIrasytiSekundes(vector <SLaikas>&,int,int); //apskaičiuoja tarpines
sekundes
      void IrasytiILentele(TTable*,TTable*,vector<SLaikas>&,AnsiString,AnsiString); //surašo
duomenis į lentelę
      void TikrintiArTeisingai(TTable*,TMemo*); //tikrina, ar teisingai įrašytos sekundės
      TDuomenuIterpimas DuomIterp;  //klasės TDuomenuIterpimas objektas
};
//---------------------------------------------------------------------------
```

IterptiSekundes.cpp

```
//---------------------------------------------------------------------------
void TIterptiSekundes::SekundziuIterpimas(TCheckListBox*check1,TCheckListBox*check2,

TCheckListBox*check3,TQuery*query,TLabel*lPas,TLabel*lRez,
                                          TTable *table1, TTable
*table2,TMemo*memo,TComboBox*combo)
{
  AnsiString tekstas;

 //--
  AnsiString DuomenuFailasAt = "";
  AnsiString DBVardasAt = "", lentelesPavAt = "";

  tekstas = "Pasirinkite, kurios lentelės duomenis norite redaguoti";
  Form1->Vardai(tekstas,DuomenuFailasAt,DBVardasAt,lentelesPavAt);
  if(DuomenuFailasAt == "")
    return;
  lPas->Caption = DuomenuFailasAt;
  //--
```

```
        AnsiString DuomenuFailasIr = "";
        AnsiString DBVardasIr = "", lentelesPavIr = "";

        tekstas = "Pasirinkite, į kurią lentelę rašyti pakeistus duomenis";
        Form1->Vardai(tekstas,DuomenuFailasIr,DBVardasIr,lentelesPavIr);
        if(DuomenuFailasIr == "")
          return;

          lRez->Caption = DuomenuFailasIr;

        if(DuomenuFailasAt == DuomenuFailasIr)
        {
         ShowMessage("Į šią lentelę įrašyti negalima!");
         return;
        }

       //--
         table1->Active = false;
         table1->DatabaseName = DBVardasAt;
         table1->TableName = lentelesPavAt;

         table2->Active = false;
         table2->DatabaseName = DBVardasIr;
         table2->TableName = lentelesPavIr;

       //--
        AnsiString uzklausa;   //pagalbinis kintamasis SQL užklausos formavimui
        AnsiString kriterijus;


        kriterijus = "Data";
        uzklausa = "select distinct " + kriterijus + " from " + lentelesPavAt + " order by " +
kriterijus;
        DuomIterp.UzpildytiDuomenuPav(check1,query,DBVardasAt,uzklausa,kriterijus);

        kriterijus = "Trumpinys";
        uzklausa = "select distinct " + kriterijus + " from " + lentelesPavAt + " order by " +
kriterijus;
        DuomIterp.UzpildytiDuomenuPav(check2,query,DBVardasAt,uzklausa,kriterijus);


        int intervalas = StrToInt(combo->Items->Strings[combo->ItemIndex]);   //intervalo dydis
minutėmis

FormuotiLaikoIntervalus(check1,check2,check3,query,table1,table2,lentelesPavAt,DBVardasAt,intervalas);

        TikrintiArTeisingai(table2,memo);

      }
      //---------------------------------------------------------------------------
      void TIterptiSekundes::FormuotiLaikoIntervalus(TCheckListBox*check1,TCheckListBox*check2,
                                                     TCheckListBox*check3,TQuery*query,
                                                     TTable*table1,TTable*table2,
                                                     AnsiString lentelesPavAt,AnsiString DBVardasAt,int
intervalas)
      {
       AnsiString pirmasLaikas,paskutinisLaikas,uzTekstas;
       AnsiString data,akcija;
       AnsiString kriterijus = "Laikas";

       //int intervalas; //intervalo dydis minutėmis

       for(int i = 0; i < check1->Items->Count; i++)
       {
        check1->ItemIndex = i;
        data = check1->Items->Strings[i];
        check2->ItemIndex = 0;
        for(int j = 0; j < check2->Items->Count; j++)
          check2->Checked[j] = false;
        for(int j = 0; j < check2->Items->Count; j++)
        {
         check2->ItemIndex = j;
         akcija = check2->Items->Strings[j];
         for(int ii = 0; ii < check3->Items->Count; ii++)
           check3->Checked[ii] = false;
         uzTekstas = "select min(" + kriterijus + "), max(" + kriterijus + ")  from " + lentelesPavAt +
" where Data ='"
                     + data + "' and Trumpinys ='" + akcija + "'";
         RastiPirmaPaskutiniLaika(query,DBVardasAt,uzTekstas,kriterijus,pirmasLaikas,paskutinisLaikas);
         if(pirmasLaikas == "")
```

```
          continue;
         FormuotiIntervalus(check3,pirmasLaikas,paskutinisLaikas,intervalas);

SkaiciuotiSekundes(check3,query,table1,table2,data,akcija,DBVardasAt,lentelesPavAt,intervalas);
         check2->Checked[j] = true;
        }
        check1->Checked[i] = true;
       }
      }
      //---------------------------------------------------------------------------
      AnsiString TIterptiSekundes::RastiPirmaPaskutiniLaika(TQuery *query,AnsiString DBVardas,
                                                AnsiString uzTekstas,AnsiString kriterijus,
                                                AnsiString &pirmasLaikas,AnsiString
&paskutinisLaikas)
      {
        AnsiString laikas;

        query->DatabaseName = DBVardas;
        query->Close();
        query->SQL->Clear();
        query->SQL->Add(uzTekstas);
        query->Open();

        pirmasLaikas = query->Fields->Fields[0]->AsString;
        paskutinisLaikas = query->Fields->Fields[1]->AsString;
        query->Close();

        return laikas;
      }
      //---------------------------------------------------------------------------
      void TIterptiSekundes::FormuotiIntervalus(TCheckListBox *check,AnsiString pradLaikas,AnsiString
pabLaikas,int intervalas)
      {
       AnsiString pglLaikas, pglMin, pglVal;
       int pglLaikasMin;
       div_t x;

       check->Clear();
       check->Items->Add(pradLaikas);
       pglLaikas = pradLaikas;
       while(pglLaikas < pabLaikas)
       {
        pglLaikasMin = LaikasMinSek(pglLaikas) + intervalas;
        x = div(pglLaikasMin,60);

        pglVal = IntToStr(x.quot);
        pglMin = IntToStr(x.rem);
        if(x.quot < 10)
          pglVal = "0" + IntToStr(x.quot);
        if(x.rem < 10)
          pglMin = "0" + IntToStr(x.rem);

        pglLaikas = pglVal + ":" + pglMin ;

        check->Items->Add(pglLaikas);
       }
       if(pglLaikas > pabLaikas)
        check->Items->Delete(check->Items->Count - 1);

      }
      //---------------------------------------------------------------------------
      int TIterptiSekundes::LaikasMinSek(AnsiString laikas)
      {
        AnsiString pglLaikas;
        int laikasMinSec;

        pglLaikas = laikas;

        if(pglLaikas.Length() == 5)
         laikasMinSec = StrToInt(pglLaikas.SubString(1,2))*60 + StrToInt(pglLaikas.SubString(4,2));

        if(pglLaikas.Length() == 8)
         laikasMinSec = StrToInt(pglLaikas.SubString(1,2))*3600 + StrToInt(pglLaikas.SubString(4,2))*60
+ StrToInt(pglLaikas.SubString(7,2));

       return laikasMinSec;
      }
      //---------------------------------------------------------------------------
      void TIterptiSekundes::SkaiciuotiSekundes(TCheckListBox *check,TQuery *query,
                                                TTable*table1,TTable*table2,
```

```
                                                AnsiString data, AnsiString akcija,
                                                AnsiString DBVardas, AnsiString lentele,
                                                int intervalas)
      {
        AnsiString uzTekstas;
        vector <SLaikas> laikai;
        vector <SLaikas> intLaikai;
        AnsiString pradLaikasInt, pabLaikasInt;   //intervalo pradinis ir galinis laikai
        int pglInd;

        AnsiString kriterijus1 = "Laikas", kriterijus2 = "Komp_laik";

        uzTekstas = "select distinct " + kriterijus1 + "," + kriterijus2 + " from "
                    + lentele + " where Data ='" + data + "' and Trumpinys = '" + akcija + "' order by "
+ kriterijus2;
        SudarytiLaikuVektoriu(query,DBVardas,uzTekstas,kriterijus1,kriterijus2,laikai);

        //--- kiekvienam intervalui
        for(int i = 0; i < check->Items->Count; i++)
        {
           check->ItemIndex = i;
           pradLaikasInt = check->Items->Strings[i];
           PabLaikasInt(pradLaikasInt,pabLaikasInt,intervalas);
           if(pabLaikasInt > laikai.back().laikasS)
            pabLaikasInt = laikai.back().laikasS;
           SekundesIntervale(table1,table2,laikai,intLaikai,pradLaikasInt,pabLaikasInt,data,akcija);

           check->Checked[i] = true;
        }


      }
      //---------------------------------------------------------------------------
      void TIterptiSekundes::PabLaikasInt(AnsiString pradLaikas, AnsiString &pabLaikas, int intervalas)
      {
        int pglLaikasMin;
        div_t x;
        AnsiString pglVal, pglMin, pglLaikas;

        pglLaikas = pradLaikas;

        pglLaikasMin = LaikasMinSek(pglLaikas) + intervalas - 1;
        x = div(pglLaikasMin,60);

        pglVal = IntToStr(x.quot);
        pglMin = IntToStr(x.rem);
        if(x.quot < 10)
          pglVal = "0" + IntToStr(x.quot);
        if(x.rem < 10)
          pglMin = "0" + IntToStr(x.rem);

        pabLaikas = pglVal + ":" + pglMin;

      }
      //---------------------------------------------------------------------------
      void TIterptiSekundes::SekundesIntervale(TTable*table1,TTable*table2,
                                               vector <SLaikas> &laikai,
                                               vector <SLaikas> &laikaiInt,
                                               AnsiString pradLaikas, AnsiString pabLaikas,
                                               AnsiString data,AnsiString akcija)
      {
        int nulSek;
        int pradIndex = -1;
        int galIndex = -1;

        int prad; //ciklo pradzia

        //--- intervalo laikų vektorius
        laikaiInt.clear();
        for(int i = 0; i < (int) laikai.size(); i++)
           if(pradLaikas <= laikai[i].laikasS && laikai[i].laikasS <= pabLaikas)
             laikaiInt.push_back(SLaikas(laikai[i].laikasS, laikai[i].laikasK,"0"));
        //---

        prad = 0;
        RastiIndeksus(laikaiInt,prad,pradIndex,galIndex);
        ApskaiciuotiIrasytiSekundes(laikaiInt,pradIndex,galIndex);

        while(galIndex < (int)laikaiInt.size() - 1)
        {
```

```
         prad = galIndex + 1;
         pradIndex = -1;
         galIndex = -1;
         RastiIndeksus(laikaiInt,prad,pradIndex,galIndex);
         ApskaiciuotiIrasytiSekundes(laikaiInt,pradIndex,galIndex);
        }
        IrasytiILentele(table1,table2,laikaiInt,data,akcija);
      }
      //---------------------------------------------------------------------------
      void TIterptiSekundes::RastiIndeksus(vector <SLaikas> laikaiInt, int prad, int &pradzia, int
&pabaiga)
      {

       int pgl1,pgl2;
       int busena; // busena = 1, kai nera langu ir aa:bb = cc:dd
                   // busena = 2, kai nera langu ir aa:bb:ss - cc:dd:ss <=59
                   //busena = 3, kai nera langu ir  aa:bb:ss - cc:dd:ss >59

       busena = 0; //neutrali pozicija

       for(int i = prad; i < (int)laikaiInt.size() - 1; i++)
       {
         if(pradzia == -1)
          pradzia = i;

         if(laikaiInt[i].laikasS < laikaiInt[i+1].laikasS) //serverio laikai skirtingi
         {
          pgl1 = LaikasMinSek(laikaiInt[i].laikasS);
          pgl2 = LaikasMinSek(laikaiInt[i+1].laikasS);
          if(pgl2 - pgl1 <= 1) //lango nėra
          {
           if(laikaiInt[i].laikasK.SubString(1,5) == laikaiInt[i+1].laikasK.SubString(1,5)) //aa:bb =
cc:dd
           {
            AnsiString a = laikaiInt[i].laikasK.SubString(1,5);
            AnsiString b = laikaiInt[i+1].laikasK.SubString(1,5);
            if(busena == 0)
              busena = 1;
            else if(busena != 1)
            {
              pabaiga = i;
              break;
            }
           }
           else  //aa:bb <> cc:dd
           {
             pgl1 = LaikasMinSek(laikaiInt[i].laikasK);
             pgl2 = LaikasMinSek(laikaiInt[i+1].laikasK);
             if(pgl2 - pgl1 <= 59) //cc:dd:ss - aa:bb:ss <= 59  (10:10:54 ir 10:11:03)
             {
              if(busena == 0)
                busena = 2;
              else if(busena != 2)
              {
                pabaiga = i;
                break;
              }
             }
             else //cc:dd:ss - aa:bb:ss > 59  (10:10:54 ir 10:11:55 arba 10:50:30 ir 10:53:29)
             {
               if(busena == 0)
                busena = 3;
               else if(busena != 3)
               {
                pabaiga = i;
                break;
               }
             }
           }
          }
          else  //langas yra
          {
              pabaiga = i;
              break;
          }
       }
         if(i == (int)laikaiInt.size() - 2)
          pabaiga = i + 1;

        } //for pabaiga
```

```
        if(prad == (int) laikaiInt.size() - 1)
        {
          if(pradzia == -1)
           pradzia = prad;

          pabaiga = prad + 1;
        }

      }
      //---------------------------------------------------------------------------
      void TIterptiSekundes::ApskaiciuotiIrasytiSekundes(vector <SLaikas> &laikaiInt, int pradIndex,int
galIndex)
      {
        int PSkirtIndex = -1;
        int pgl1,pgl2,pglSek, a,aa;
        int stulpKmax = -1;
        int stulpDmin = 100000;
        int skirtumas = 100000;
        int var = 0;
        int atskaitaS = 0;
        int atskaitaK = 0;
        int sek = 0;
        int nulSek = 0;
        AnsiString sekundes;
        AnsiString pglSekundes;
        int vienodiLaikai = 0;



        if(pradIndex == galIndex)
          laikaiInt[pradIndex].laikasP = laikaiInt[pradIndex].laikasS +
laikaiInt[pradIndex].laikasK.SubString(6,3);
        else
         for(int i = pradIndex; i < galIndex; i++)
           if(laikaiInt[i].laikasS < laikaiInt[i+1].laikasS) //serverio laikai skirtingi
           {
             if(PSkirtIndex == -1)
               PSkirtIndex = i + 1;

             if(laikaiInt[i].laikasK.SubString(1,5) == laikaiInt[i+1].laikasK.SubString(1,5)) //aa:bb =
cc:dd
             {
               var = 1;
               pgl1 = StrToInt(laikaiInt[i].laikasK.SubString(7,2));
               pgl2 = StrToInt(laikaiInt[i+1].laikasK.SubString(7,2));
               if(stulpKmax < pgl1)
                 stulpKmax = pgl1;
               if(stulpDmin > pgl2)
                 stulpDmin = pgl2;
             }
             else  //aa:bb <> cc:dd
             {
              pgl1 = LaikasMinSek(laikaiInt[i].laikasK);
              pgl2 = LaikasMinSek(laikaiInt[i+1].laikasK);
              if(pgl2 - pgl1 <= 59) //cc:dd:ss - aa:bb:ss <= 59  (10:10:54 ir 10:11:03)
              {
               var = 2;
               pgl1 = StrToInt(laikaiInt[i].laikasK.SubString(7,2));
               pgl2 = StrToInt(laikaiInt[i+1].laikasK.SubString(7,2));
               if(stulpKmax < pgl1)
                 stulpKmax = pgl1;
               if(stulpDmin > pgl2)
                 stulpDmin = pgl2;
              }
              else
              {
                var = 3;
                if(skirtumas > pgl2 - pgl1)
                 skirtumas = pgl2 - pgl1;
              }
             }
           }
          else //serverio laikai vienodi
           vienodiLaikai++;

         if(vienodiLaikai == galIndex - pradIndex && vienodiLaikai != 0)
          var = 4;
```

```
         switch(var)
         {
          case 1:
             nulSek = floor((stulpDmin - stulpKmax)/2);
             atskaitaS = LaikasMinSek(laikaiInt[PSkirtIndex].laikasS + ":00");
             if(nulSek + stulpKmax < 10)
              pglSekundes = "0" + IntToStr(nulSek + stulpKmax);
             else
               pglSekundes = IntToStr(nulSek + stulpKmax);

             atskaitaK = LaikasMinSek(laikaiInt[PSkirtIndex].laikasK.SubString(1,5) + ":" +
pglSekundes);
             for(int i = pradIndex; i <= galIndex; i++)
             {
              sek = LaikasMinSek(laikaiInt[i].laikasK) - atskaitaK;
               a = div(atskaitaS + sek,60).rem;

              if(a < 10)
               sekundes = "0" + IntToStr(a);
              else
               sekundes = IntToStr(a);

              laikaiInt[i].laikasP = laikaiInt[i].laikasS + ":" + sekundes;

              if(a == 0 && i != 0)
               if(laikaiInt[i - 1].laikasP > laikaiInt[i].laikasP)
                 laikaiInt[i].laikasP = laikaiInt[i].laikasS + ":59";

             }
            break;
          case 2:
             nulSek = floor((stulpDmin + 60 - stulpKmax)/2);
             atskaitaS = LaikasMinSek(laikaiInt[PSkirtIndex].laikasS + ":00");
              pglSek = stulpKmax + nulSek;
             if(pglSek < 10)
              pglSekundes = "0" + IntToStr(pglSek);
             else
               pglSekundes = IntToStr(pglSek);

             if(pglSek < 60)
              atskaitaK = LaikasMinSek(laikaiInt[PSkirtIndex - 1].laikasK.SubString(1,5) + ":"+
pglSekundes);
             else
             {
              pglSek = pglSek - 60;
              atskaitaK = LaikasMinSek(laikaiInt[PSkirtIndex].laikasK.SubString(1,5) + ":"
+pglSekundes);
             }

             for(int i = pradIndex; i <= galIndex; i++)
             {
              sek = LaikasMinSek(laikaiInt[i].laikasK) - atskaitaK;
              aa = div(atskaitaS + sek,60).rem;

              if(aa < 10)
               sekundes = "0" + IntToStr(aa);
              else
               sekundes = IntToStr(aa);

              laikaiInt[i].laikasP = laikaiInt[i].laikasS + ":" + sekundes;

              if(aa == 0 && i != 0)
               if(laikaiInt[i - 1].laikasP > laikaiInt[i].laikasP)
                 laikaiInt[i].laikasP = laikaiInt[i].laikasS + ":59";

             }
           break;

        case 3:
           for(int i = pradIndex; i <= galIndex; i++)
            laikaiInt[i].laikasP = laikaiInt[i].laikasS + laikaiInt[i].laikasK.SubString(6,3);
        break;

        case 4:
           laikaiInt[pradIndex].laikasP = laikaiInt[pradIndex].laikasS + ":00";
           for(int i = pradIndex + 1; i <= galIndex; i++)
           {
            sek = LaikasMinSek(laikaiInt[i].laikasK) - LaikasMinSek(laikaiInt[pradIndex].laikasK);
            if(sek < 10)
```

```
            sekundes = "0" + IntToStr(sek);
           else
            sekundes = IntToStr(sek);

           laikaiInt[i].laikasP = laikaiInt[i].laikasS + ":" + sekundes;
          }

         break;
       }
      }
      //---------------------------------------------------------------------------
      void TIterptiSekundes::TikrintiArTeisingai(TTable*table,TMemo*memo)
      {
       AnsiString laikas1,data1,akcija1;
       AnsiString laikas2,data2,akcija2;
       memo->Clear();

       table->First();
       table->Next();
       while(!table->Eof)
       {
        data2 = table->FieldByName(table->Fields->Fields[0]->FieldName)->AsString;
        akcija2 = table->FieldByName(table->Fields->Fields[1]->FieldName)->AsString;
        laikas2 = table->FieldByName(table->Fields->Fields[2]->FieldName)->AsString;

        table->Prior();
        data1 = table->FieldByName(table->Fields->Fields[0]->FieldName)->AsString;
        akcija1 = table->FieldByName(table->Fields->Fields[1]->FieldName)->AsString;
        laikas1 = table->FieldByName(table->Fields->Fields[2]->FieldName)->AsString;
        if(laikas2 < laikas1 && data1 == data2 && akcija1 == akcija2)
         memo->Lines->Add(data1 + " " + akcija1 + " " +laikas1 + " " + laikas2);

        table->Next();
        table->Next();
       }


      }
      //---------------------------------------------------------------------------
```

DuomenuIterpimas.h

```
      //---------------------------------------------------------------------------
      struct SLaikaiPP
      {
        SLaikaiPP(AnsiString prad, AnsiString pab)
        { pradLaikas = prad; pabLaikas = pab; }
        AnsiString pradLaikas, pabLaikas;
      };

      struct SSandoriai
      {
       SSandoriai(AnsiString d, AnsiString a, AnsiString l, AnsiString ll, double ka,AnsiString v,
double ki)
       {data = d; akcija = a; laikas = l; laikasPriesSand = ll; kaina = ka; val = v; kiekis = ki;}
       AnsiString data, akcija,laikas, laikasPriesSand,val;
       double kaina;
       int kiekis;
      };

      struct SRGLentele
      {
       SRGLentele(AnsiString d, AnsiString t, AnsiString prL, AnsiString pbL,
                  int piKi, double piKa, AnsiString piV, double paKa, AnsiString paV, int paKi)
       {data = d; akcija = t; pradLaikas = prL; pabLaikas = pbL; pirkKiekis = piKi; pirkKaina = piKa;
        pirkVal = piV; pardKaina = paKa; pardVal = paV; pardKiekis = paKi;}

        AnsiString data,akcija,pradLaikas,pabLaikas,pirkVal,pardVal;
        double pirkKaina,pardKaina;
        int pirkKiekis,pardKiekis;
      };

      struct SRGStulpelis
      {
        SRGStulpelis(double k, AnsiString v, int ki)
        {kaina = k; valiuta = v; kiekis = ki;}
        double kaina;
```

```
        AnsiString valiuta;
        int kiekis;
      };
      //---------------------------------------------------------------------------
      class TDuomenuIterpimas
      {
       public:

        void UzpildytiDuomenuPav(TCheckListBox *, TQuery *, AnsiString, AnsiString,AnsiString);
//užpildo dumenų pavadinimo lauką
        void Vardai(AnsiString, AnsiString &,AnsiString &);//randa duomenų bazės ir lentelės vardus
        void UzpildytiVektoriu(TQuery*, AnsiString,AnsiString, AnsiString, vector <SLaikaiPP>&,vector
<SSandoriai>&,bool); //užpildo nurodytą vektorių duomenimis
        void AtrinktiLaikus(vector <SLaikaiPP> &,vector <SLaikaiPP> &,vector <SLaikaiPP> &,vector
<SSandoriai> &,
                                              vector <SSandoriai> &,AnsiString,AnsiString);//pagal tam
tikrą kriterijų atrenka laikus

        void
Iterpti(TTable*,TTable*,TTable*,TTable*,TMemo*,vector<SLaikaiPP>&,vector<SLaikaiPP>&,vector<SSandoriai>
&,AnsiString,AnsiString,AnsiString&,AnsiString&); //pagrindinis metodas duomenų įterpimui
        void
Iterpti212(TTable*,TTable*,AnsiString,double,int,bool&,AnsiString&,AnsiString&,AnsiString&); //metodas
duomenims įterpti, kai atitinkamo laiko rinkos gylio lentelėje yra 1 nuotrauka  (1 ir 2 algoritmai)
        void Iterpti2(TTable*,TTable*,AnsiString,double,int,bool&,AnsiString
&,AnsiString&,AnsiString&); //metodas duomenims įterpti, kai atitinkamo laiko rinkos gylio lentelėje
yra 1 nuotrauka
        void
Iterpti3(TTable*,TTable*,TTable*,double,int,bool&,bool&,AnsiString&,AnsiString&,AnsiString&); //metodas
duomenims įterpti, kai atitinkamo laiko rinkos gylio lentelėje yra daugiau nei 1 nuotrauka


        void KurKaina(TTable*,int&,bool&,bool&,AnsiString&,AnsiString&,double); //ieško kuriame
stulpelyje yra kaina
        void ArYraKaina(TTable*,AnsiString,AnsiString,double,int&,bool&);//tikrina, ar yra lentelės
nurodytame stulpelyje kaina
        void SandorisNeivyko(TTable*,TTable*,AnsiString,AnsiString,bool&,AnsiString&,AnsiString&);
//jei sandoris neįvyko ir jei reikia, įrašo lentelę
        void
SandorisIvyko(TTable*,TTable*,AnsiString,AnsiString,int,AnsiString,bool&,bool&,bool,AnsiString&,AnsiStr
ing&,AnsiString&); //veiksmai, kai sandoris įvyko
        void
KainosNera(TTable*,TTable*,AnsiString,double,int,bool&,bool&,bool&,AnsiString&,AnsiString&,AnsiString&)
; //veiksmai, kai lentelėje nėra kainos

        void PerkeltiLentele(TTable*,TTable*,AnsiString,bool); //įrašo lentelę į duomenų bazę
        void IrasytiLentele(vector<SRGLentele>&,TTable*); //įrašo lentelę į duomenų bazę
        void PakeistiIrasa(TTable*,vector <SRGLentele> &,AnsiString,int,bool); //pakeičia lentelėje
įrašą nauju įrašu
        void IterptiIrasa(TTable*,vector <SRGLentele> &,AnsiString, double, int , bool );// įrašo į
naują lentelę į nurodytą stulpelį nurodytą įrašą
        void IstrintiIrasa(TTable*,vector <SRGLentele> &,AnsiString,bool); //ištrina iš lentelės
nurodytą įrašą

        void
NagrinetiLentele(TTable*,TTable*,AnsiString,double,int,AnsiString,bool,bool&,AnsiString&,AnsiString&,An
siString&); //tikrina, kuriam lentelės stulpelyje reikia iterpti/istrinti ar nieko nedaryti su įrašu
        void FiltruotiLentele(TTable*,AnsiString); //filtruoja lentelę pagal nurodytus kriterijus
        int LygintiPradPabLaikus(AnsiString,AnsiString,AnsiString); //palygina du laikus
        AnsiString PridetiSekundes(AnsiString,int); //prie nurodyto laiko prideda sekundes
        bool LygintiLaikus(AnsiString,AnsiString); //lygina dvejus laikus (formatas: "11:39")
        void LygintiIrasus(TQuery*,TTable*,TTable*,TTable*); //lygina ar lentelės įrašai yra vienodi

        void
IrasytiLikusiasLenteles(TTable*,TTable*,vector<SLaikaiPP>&,AnsiString,AnsiString,AnsiString&,AnsiString
&); //į DB įrašo likusias lenteles
        void AtrinktiLaikus2(vector <SLaikaiPP> &,vector <SLaikaiPP> &,AnsiString); //atrenka laikus
        void IrasytiSandLentele(vector<SSandoriai>&,TTable*); //įrašo sandorių lentelę


        TDuomSutraukimas DuomSut;   //klasės TDuomSutraukimas objektas

      };
      //---------------------------------------------------------------------------
```

DuomenuIterpimas.cpp

```
      //---------------------------------------------------------------------------
      void TDuomenuIterpimas::UzpildytiDuomenuPav(TCheckListBox *langas, TQuery *query,
                                AnsiString DBVardas, AnsiString uzklausa,
```

```
                                AnsiString kriterijus)
{

  query->DatabaseName = DBVardas;

  langas->Items->Clear();

  query->Close();
  query->SQL->Clear();
  query->SQL->Add(uzklausa);
  query->Open();

  while(!query->Eof)
   {
    langas->Items->Add(query->FieldByName(kriterijus)->AsString);
    query->Next();
   }

  query->Close();
}
//---------------------------------------------------------------------------
void TDuomenuIterpimas::Vardai(AnsiString failas, AnsiString &DBVardas,AnsiString &lentelesPav)
{

 int k = failas.LastDelimiter("\\");   //paskutinio "\" vieta
 DBVardas = failas.SubString(1, k - 1);
 lentelesPav = failas.SubString(k + 1, failas.Length() - k + 1);
}
//---------------------------------------------------------------------------
void TDuomenuIterpimas::AtrinktiLaikus(vector <SLaikaiPP> &laikai,
                                       vector <SLaikaiPP> &atrinktiLaikai,
                                       vector <SLaikaiPP> &likeLaikai,
                                       vector <SSandoriai> &sandoriai,
                                       vector <SSandoriai> &atrinktiSandoriai,
                                       AnsiString laikas,AnsiString laikasP)
{
    atrinktiLaikai.clear();
    likeLaikai.clear();

    for(int i = 0; i < (int) laikai.size(); i++)
     if(laikai[i].pradLaikas.SubString(1,5) <= laikas && laikas <=
laikai[i].pabLaikas.SubString(1,5))
         atrinktiLaikai.push_back(laikai[i]);
      else
        if(laikai[i].pradLaikas.SubString(1,5) < laikas
          && laikai[i].pabLaikas.SubString(1,5) < laikas
          &&laikai[i].pradLaikas.SubString(1,5) > laikasP
          && laikai[i].pabLaikas.SubString(1,5) > laikasP)
            likeLaikai.push_back(laikai[i]);

   atrinktiSandoriai.clear();
   for(int i = sandoriai.size() - 1; i >= 0; i--)
    if(sandoriai[i].laikas == laikas)
       atrinktiSandoriai.push_back(sandoriai[i]);
}
//---------------------------------------------------------------------------
void TDuomenuIterpimas::Iterpti(TTable*table1,TTable*table2,TTable*table3,TTable*table4,
                                TMemo*langas,
                                vector<SLaikaiPP>&atrinktiLaikai,
                                vector<SLaikaiPP>&likeLaikai,
                                vector<SSandoriai>&atrinktiSandoriai,
                                AnsiString data, AnsiString akcija,
                                AnsiString &paskLaikasPr,AnsiString &paskLaikasPb)
{
  AnsiString stulp1, stulp2,filtras,filtras1,filtras2,pradLaikasP,pabLaikasP,akcijaP,dataP;
  int algoritmas;
  bool arIrasyta = false;
  bool arIvykoSandoris = false;
  vector<SRGLentele> tmpVektorius;
  AnsiString laikasPriesSand = "";

try{
  table1->Active = true;
  stulp1 = table1->Fields->Fields[2]->FieldName;
  stulp2 = table1->Fields->Fields[3]->FieldName;
  akcijaP = table1->Fields->Fields[1]->FieldName;
  dataP = table1->Fields->Fields[0]->FieldName;
  pradLaikasP = stulp1;
  pabLaikasP = stulp2;
```

```
        //įrašom lenteles tų laikų, kuomet nėra sandorių
        for(int i = 0; i < (int) likeLaikai.size(); i++)
        {

          if(likeLaikai[i].pradLaikas != paskLaikasPr && likeLaikai[i].pabLaikas != paskLaikasPb
             && paskLaikasPb < likeLaikai[i].pradLaikas)
          {
           filtras = dataP + " ='" + data + "' and " + akcijaP + " ='" + akcija + "' and "  + stulp1 +
" = '" + likeLaikai[i].pradLaikas + "' and "
                    + stulp2 + " = '" + likeLaikai[i].pabLaikas + "'";

           FiltruotiLentele(table1,filtras);
           PerkeltiLentele(table1,table2,"",false);
           paskLaikasPr = likeLaikai[i].pradLaikas;
           paskLaikasPb = likeLaikai[i].pabLaikas;
          }
         }

        //sudarom rinkos gylio lentelių "nuotraukas" prieš ir po sandorių
        if(atrinktiLaikai.size() == 0)
        {
         for(int i = 0; i < (int) atrinktiSandoriai.size(); i++)
         {
          filtras = dataP + " ='" + data + "' and " + akcijaP + " ='" + akcija + "' and " + stulp1 + "
= '" +paskLaikasPr + "' and "
                    + stulp2 + " = '" + paskLaikasPb + "'";
          FiltruotiLentele(table2,filtras);

          if(i == 0)
          {
           AnsiString pglLaikas = atrinktiSandoriai[0].laikas + ":00";
           PerkeltiLentele(table2,table3,pglLaikas,true);
          }
          else
            PerkeltiLentele(table2,table3,"",false);
          table2->Filtered = false;
          table2->Filter = "";

          table3->Active = true;

          Iterpti2(table3,table2,atrinktiSandoriai[i].laikas,atrinktiSandoriai[i].kaina,
                   atrinktiSandoriai[i].kiekis,arIrasyta,paskLaikasPr,paskLaikasPb,laikasPriesSand);

          table3->Active = false;
          table3->EmptyTable();
          if(!arIrasyta)
          {
           for(int j = i; j < (int) atrinktiSandoriai.size(); j++)
            langas->Lines->Add(data + " " + akcija + " " + atrinktiSandoriai[j].laikas
                               + " " + atrinktiSandoriai[j].kaina+ " " + atrinktiSandoriai[j].kiekis);
           return;
          }
          atrinktiSandoriai[i].laikasPriesSand = laikasPriesSand;
          laikasPriesSand = "";
         }
        }
        else
          if(atrinktiLaikai.size() == 1)
          {
           for(int i = 0; i < (int) atrinktiSandoriai.size(); i++)
           {
            if(i == 0)
            {
             filtras = dataP + " ='" + data + "' and " + akcijaP + " ='" + akcija + "' and " + stulp1 +
" = '" + atrinktiLaikai[0].pradLaikas + "' and "
                      + stulp2 + " = '" + atrinktiLaikai[0].pabLaikas + "'";
             FiltruotiLentele(table1,filtras);
             PerkeltiLentele(table1,table3,"",false);
            }
            else
            {
             filtras = dataP + " ='" + data + "' and " + akcijaP + " ='" + akcija + "' and "  + stulp1
+ " = '" +paskLaikasPr + "' and "
                      + stulp2 + " = '" + paskLaikasPb + "'";
             FiltruotiLentele(table2,filtras);
             PerkeltiLentele(table2,table3,"",false);
             table2->Filtered = false;
             table2->Filter = "";
            }
```

```
            table3->Active = true;

            Iterpti2(table3,table2,atrinktiSandoriai[i].laikas,atrinktiSandoriai[i].kaina,
                     atrinktiSandoriai[i].kiekis,arIrasyta,paskLaikasPr,paskLaikasPb,laikasPriesSand);

            table3->Active = false;
            table3->EmptyTable();
            if(!arIrasyta)
            {
             for(int j = i; j < (int) atrinktiSandoriai.size(); j++)
                langas->Lines->Add(data + " " + akcija + " " + atrinktiSandoriai[j].laikas
                                   + " " + atrinktiSandoriai[j].kaina+ " " +
atrinktiSandoriai[j].kiekis);
             return;
            }
            atrinktiSandoriai[i].laikasPriesSand = laikasPriesSand;
            laikasPriesSand = "";
           }

           int algoritmas = LygintiPradPabLaikus(table1->FieldByName(pradLaikasP)->AsString,
                                                 table1->FieldByName(pabLaikasP)-
>AsString,atrinktiSandoriai[0].laikas);
           if(algoritmas == 3)
           {
            filtras1 = dataP + " ='" + data + "' and " + akcijaP + " ='" + akcija + "' and " + stulp1 +
" = '" + atrinktiLaikai[0].pradLaikas + "' and "
                           + stulp2 + " = '" + atrinktiLaikai[0].pabLaikas + "'";

            FiltruotiLentele(table1,filtras1);
            if(paskLaikasPb < atrinktiLaikai[0].pradLaikas)
            {
              PerkeltiLentele(table1,table2,"",false);
              paskLaikasPr = atrinktiLaikai[0].pradLaikas;
              paskLaikasPb = atrinktiLaikai[0].pabLaikas;
            }
            else
             langas->Lines->Add("RGylio lentle " + data + " " + akcija + " " +
atrinktiLaikai[0].pradLaikas
                                + "-" + atrinktiLaikai[0].pabLaikas + " neįrašyta");
           }
           else if(algoritmas == 4)
           {
            filtras1 = dataP + " ='" + data + "' and " + akcijaP + " ='" + akcija + "' and "  + stulp1
+ " = '" + atrinktiLaikai[0].pradLaikas + "' and "
                           + stulp2 + " = '" + atrinktiLaikai[0].pabLaikas + "'";

            FiltruotiLentele(table1,filtras1);
            AnsiString pglLaikas = PridetiSekundes(paskLaikasPb,1);

            if(pglLaikas <= atrinktiLaikai[0].pabLaikas)
            {

             //perkeliam lentele
             table1->Active =  true;
             table1->First();
             table2->Active = true;
             table2->Next();
             while(!table1->Eof)
             {
              table2->Append();
              for(int ii = 0; ii < table1->FieldCount; ii++)
               if(ii == 2)
                 table2->FieldValues[table2->Fields->Fields[ii]->FieldName] = pglLaikas;
               else
                 table2->FieldValues[table2->Fields->Fields[ii]->FieldName]
                      = table1->FieldValues[table1->Fields->Fields[ii]->FieldName];
              table2->Post();
              table1->Next();
              table2->Next();
             }
             table2->Active = false;
             table1->First();

             paskLaikasPr = pglLaikas;
             paskLaikasPb = atrinktiLaikai[0].pabLaikas;

            }
            else
              langas->Lines->Add("Rinkos gylio lentelė " + data + " " + akcija + " " +
atrinktiLaikai[0].pradLaikas
```

```
                                + " " + atrinktiLaikai[0].pabLaikas+ " neįrašyta");
          }
         }
         else
          if (atrinktiLaikai.size () > 1)
          {
           arIvykoSandoris = false;
           int met;
           for(int i = 0; i < (int) atrinktiSandoriai.size(); i++)
            for(int j = 0; j < (int) atrinktiLaikai.size(); j++)
            {
             if((i == 0 && j == 0) || (!arIvykoSandoris && atrinktiLaikai.size() != 1))
             {
              filtras1 = dataP + " ='" + data + "' and " + akcijaP + " ='" + akcija + "' and "
+stulp1 + " = '" + atrinktiLaikai[j].pradLaikas + "' and "
                       + stulp2 + " = '" + atrinktiLaikai[j].pabLaikas + "'";
              filtras2 = dataP + " ='" + data + "' and " + akcijaP + " ='" + akcija + "' and "  +
stulp1 + " = '" + atrinktiLaikai[j + 1].pradLaikas + "' and "
                         + stulp2 + " = '" + atrinktiLaikai[j + 1].pabLaikas + "'";
              FiltruotiLentele(table1,filtras1);
              PerkeltiLentele(table1,table3,"",false);
              met = 1;
             }
             else
             {
              filtras1 = dataP + " ='" + data + "' and " + akcijaP + " ='" + akcija + "' and " +
stulp1 + " = '" + paskLaikasPr + "' and "
                       + stulp2 + " = '" + paskLaikasPb + "'";
              filtras2 = dataP + " ='" + data + "' and " + akcijaP + " ='" + akcija + "' and "  +
stulp1 + " = '" + atrinktiLaikai[j].pradLaikas + "' and "
                       + stulp2 + " = '" + atrinktiLaikai[j].pabLaikas + "'";
              FiltruotiLentele(table2,filtras1);
              PerkeltiLentele(table2,table3,"",false);
              table2->Filtered = false;
              table2->Filter = "";
              met = 2;
             }

             FiltruotiLentele(table1,filtras2);
             PerkeltiLentele(table1,table4,"",false);

             table3->Active = true;
             table4->Active = true;

             Iterpti3(table3,table4,table2,atrinktiSandoriai[i].kaina,

atrinktiSandoriai[i].kiekis,arIvykoSandoris,arIrasyta,paskLaikasPr,paskLaikasPb,laikasPriesSand);

             table3->Active = false;
             table3->EmptyTable();
             table4->Active = false;
             table4->EmptyTable();

             if(met == 1)
             {
              atrinktiLaikai.erase(atrinktiLaikai.begin() + j);
              j = j - 1;
             }

             if(arIvykoSandoris)
             {
              atrinktiSandoriai[i].laikasPriesSand = laikasPriesSand;
              laikasPriesSand = "";
              break;
             }
             else
              if(atrinktiLaikai.size() == 1)
              {
               //nagrinejam kaip viena lentele
               for(int k = i; k < (int) atrinktiSandoriai.size(); k++)
               {
                if(k == i)
                {
                 filtras = dataP + " ='" + data + "' and " + akcijaP + " ='" + akcija + "' and " +
stulp1 + " = '" + atrinktiLaikai[0].pradLaikas + "' and "
                          + stulp2 + " = '" + atrinktiLaikai[0].pabLaikas + "'";
                 FiltruotiLentele(table1,filtras);
                 PerkeltiLentele(table1,table3,"",false);
                }
                else
```

```
                {
                 filtras = dataP + " ='" + data + "' and " + akcijaP + " ='" + akcija + "' and " +
stulp1 + " = '" +paskLaikasPr + "' and "
                          + stulp2 + " = '" + paskLaikasPb + "'";
                 FiltruotiLentele(table2,filtras);
                 PerkeltiLentele(table2,table3,"",false);
                 table2->Filtered = false;
                 table2->Filter = "";
                }
                table3->Active = true;
                Iterpti2(table3,table2,atrinktiSandoriai[k].laikas,atrinktiSandoriai[k].kaina,

atrinktiSandoriai[k].kiekis,arIrasyta,paskLaikasPr,paskLaikasPb,laikasPriesSand);

                table3->Active = false;
                table3->EmptyTable();

                if(!arIrasyta)
                {
                 for(int jj = k; jj < (int) atrinktiSandoriai.size(); jj++)
                   langas->Lines->Add(data + " " + akcija + " " + atrinktiSandoriai[jj].laikas
                                      + " " + atrinktiSandoriai[jj].kaina+ " " +
atrinktiSandoriai[jj].kiekis);

                 int algoritmas = LygintiPradPabLaikus(table1->FieldByName(pradLaikasP)->AsString,
                                        table1->FieldByName(pabLaikasP)-
>AsString,atrinktiSandoriai[0].laikas);
                 if(algoritmas == 3)
                 {
                  filtras1 = dataP + " ='" + data + "' and " + akcijaP + " ='" + akcija + "' and "  +
stulp1 + " = '" + atrinktiLaikai[0].pradLaikas + "' and "
                               + stulp2 + " = '" + atrinktiLaikai[0].pabLaikas + "'";

                  FiltruotiLentele(table1,filtras1);
                  if(paskLaikasPb < atrinktiLaikai[0].pradLaikas && paskLaikasPr !=
atrinktiLaikai[0].pradLaikas && paskLaikasPb != atrinktiLaikai[0].pabLaikas)
                  {
                   PerkeltiLentele(table1,table2,"",false);
                   paskLaikasPr = atrinktiLaikai[0].pradLaikas;
                   paskLaikasPb = atrinktiLaikai[0].pabLaikas;
                   atrinktiLaikai.erase(atrinktiLaikai.begin());
                  }
                  else
                   langas->Lines->Add("RGylio lentle " + data + " " + akcija + " " +
atrinktiLaikai[0].pradLaikas
                                      + "-" + atrinktiLaikai[0].pabLaikas + " neįrašyta");
                 }




                 return;
                }

                atrinktiSandoriai[k].laikasPriesSand = laikasPriesSand;
                laikasPriesSand = "";

                if(k == (int) atrinktiSandoriai.size() - 1)
                {
                 int algoritmas = LygintiPradPabLaikus(table1->FieldByName(pradLaikasP)->AsString,
                                        table1->FieldByName(pabLaikasP)-
>AsString,atrinktiSandoriai[0].laikas);
                 if(algoritmas == 3)
                 {
                  filtras1 = dataP + " ='" + data + "' and " + akcijaP + " ='" + akcija + "' and "  +
stulp1 + " = '" + atrinktiLaikai[0].pradLaikas + "' and "
                               + stulp2 + " = '" + atrinktiLaikai[0].pabLaikas + "'";

                  FiltruotiLentele(table1,filtras1);
                  if(paskLaikasPb < atrinktiLaikai[0].pradLaikas && paskLaikasPr !=
atrinktiLaikai[0].pradLaikas && paskLaikasPb != atrinktiLaikai[0].pabLaikas)
                  {
                   PerkeltiLentele(table1,table2,"",false);
                   paskLaikasPr = atrinktiLaikai[0].pradLaikas;
                   paskLaikasPb = atrinktiLaikai[0].pabLaikas;
                   atrinktiLaikai.erase(atrinktiLaikai.begin());
                  }
                  else
                   langas->Lines->Add("RGylio lentle " + data + " " + akcija + " " +
atrinktiLaikai[0].pradLaikas
```

```
                                    + "-" + atrinktiLaikai[0].pabLaikas + " neįrašyta");
                  }
                 }
                }
                i = atrinktiSandoriai.size();
               }
               else
                if(met == 2)
                 j = -1;
            }
            for(int k = 0; k < (int) atrinktiLaikai.size(); k++)
            {
             filtras1 = dataP + " ='" + data + "' and " + akcijaP + " ='" + akcija + "' and " + stulp1
+ " = '" + atrinktiLaikai[k].pradLaikas + "' and "
                      + stulp2 + " = '" + atrinktiLaikai[k].pabLaikas + "'";
             if(atrinktiLaikai[k].pradLaikas != paskLaikasPr && atrinktiLaikai[k].pabLaikas !=
paskLaikasPb
                && paskLaikasPb < atrinktiLaikai[k].pradLaikas )
             {
              FiltruotiLentele(table1,filtras1);
              PerkeltiLentele(table1,table2,"",false);
              paskLaikasPr = atrinktiLaikai[k].pradLaikas;
              paskLaikasPb = atrinktiLaikai[k].pabLaikas;

             }
            }
           }
      }
      catch(...)
      {
        langas->Lines->Add("Įvyko klaida: " + data + " " + akcija);
      }
      }
      //---------------------------------------------------------------------------
      void TDuomenuIterpimas::Iterpti2(TTable*table1,TTable*table2,
                                       AnsiString laikasS, double kainaS, int kiekisS,
                                       bool &arIrasyta,AnsiString &paskLaikasPr, AnsiString
&paskLaikasPb,
                                       AnsiString&laikasPriesSand)
      {
       int algoritmas;
       AnsiString pradLaikasP = table1->Fields->Fields[2]->FieldName;
       AnsiString pabLaikasP = table1->Fields->Fields[3]->FieldName;
       AnsiString pirkKiekisP = table1->Fields->Fields[4]->FieldName;
       AnsiString laikas;
       vector <SRGLentele> naujaLentele;

       algoritmas = LygintiPradPabLaikus(table1->FieldByName(pradLaikasP)->AsString,
                                         table1->FieldByName(pabLaikasP)->AsString,laikasS);

       if(algoritmas == 1 || algoritmas == 2)

Iterpti212(table1,table2,laikasS,kainaS,kiekisS,arIrasyta,paskLaikasPr,paskLaikasPb,laikasPriesSand);
       else
         if(algoritmas == 3)
         {
          AnsiString pglLaikas1 = paskLaikasPb.SubString(1,5);
          AnsiString senasLaikas = table1->FieldByName(pradLaikasP)->AsString;
          AnsiString pglLaikas2 = senasLaikas.SubString(1,5);
          if(pglLaikas1 == pglLaikas2)
            laikas = PridetiSekundes(paskLaikasPb,1);
          else
            laikas = pglLaikas2 + ":00";

          table1->First();
          int pglKiekis = table1->FieldByName(pirkKiekisP)->AsInteger;
          PakeistiIrasa(table1,naujaLentele,laikas,pglKiekis,false);
          table1->Active = false;
          table1->EmptyTable();
          if(laikas != paskLaikasPr && laikas > paskLaikasPb && laikas < senasLaikas)
          {
           IrasytiLentele(naujaLentele,table1);
           PerkeltiLentele(table1,table2,"",false);
           paskLaikasPr = laikas;
           paskLaikasPb = laikas;

Iterpti212(table1,table2,laikasS,kainaS,kiekisS,arIrasyta,paskLaikasPr,paskLaikasPb,laikasPriesSand);
          }
         }
         else
```

```
         {
          AnsiString pglLaikasL1 = table1->FieldByName(pradLaikasP)->AsString;
          AnsiString pglLaikasL2 = pglLaikasL1.SubString(1,6) + "59";
          if(pglLaikasL1 > paskLaikasPb && pglLaikasL1 != paskLaikasPr && pglLaikasL2 != paskLaikasPb)
          {

           //perkeliam lentele
           table1->Active =  true;
           table1->First();
           table2->Active = true;
           table2->Next();
           while(!table1->Eof)
           {
            table2->Append();
            for(int i = 0; i < table1->FieldCount; i++)
            if(i == 3)
              table2->FieldValues[table2->Fields->Fields[i]->FieldName]
                   = pglLaikasL2;
            else
              table2->FieldValues[table2->Fields->Fields[i]->FieldName]
                    = table1->FieldValues[table1->Fields->Fields[i]->FieldName];
            table2->Post();
            table1->Next();
            table2->Next();
           }
           table2->Active = false;
           table1->First();

           paskLaikasPr = pglLaikasL1;
           paskLaikasPb = pglLaikasL2;

           laikas = laikasS + ":00";
           int pglKiekis = table1->FieldByName(pirkKiekisP)->AsInteger;
           PakeistiIrasa(table1,naujaLentele,laikas,pglKiekis,false);
           table1->Active = false;
           table1->EmptyTable();
           if(laikas != paskLaikasPr && laikas > paskLaikasPb)
           {
            IrasytiLentele(naujaLentele,table1);
            PerkeltiLentele(table1,table2,"",false);
            paskLaikasPr = laikas;
            paskLaikasPb = laikas;

Iterpti212(table1,table2,laikasS,kainaS,kiekisS,arIrasyta,paskLaikasPr,paskLaikasPb,laikasPriesSand);
           }
          }
         }
       }
       //---------------------------------------------------------------------------
       void TDuomenuIterpimas::Iterpti3(TTable*table1,TTable*table2,TTable*table3,
                                        double kainaS,int kiekisS,bool &arIvykoSandoris,bool &arIrasyta,
                                        AnsiString &paskLaikasPr,AnsiString
&paskLaikasPb,AnsiString&laikasPriesSand)
       {
        int index = 0;
        AnsiString pavKiekis,pavKaina;
        bool pard = false;
        bool pirk = false;
        AnsiString pradLaikasP = table1->Fields->Fields[2]->FieldName;
        AnsiString pabLaikasP = table1->Fields->Fields[3]->FieldName;
        bool arYra = false;
        int pglKiekis,pglSkirtumas;
        AnsiString pglLaikas;


        KurKaina(table1,index,pirk,pard,pavKiekis,pavKaina,kainaS);
        if(!pirk && !pard) //kainos nera
        {

SandorisNeivyko(table1,table3,pradLaikasP,pabLaikasP,arIvykoSandoris,paskLaikasPr,paskLaikasPb);
         arIvykoSandoris = false;
         return;
        }
        else   //kaina yra
         if(index != 1)  //kaina ne pirma eiluteje
         {

SandorisNeivyko(table1,table3,pradLaikasP,pabLaikasP,arIvykoSandoris,paskLaikasPr,paskLaikasPb);
          return;
         }
```

```
        else          //kaina pirma eiluteje
        {
         if(kiekisS <= table1->FieldByName(pavKiekis)->AsInteger)
         {
          table2->First();
          if(table2->FieldByName(pavKaina)->AsFloat == table1->FieldByName(pavKaina)->AsFloat) //pirmos
kainos lygios
          {
           if(table2->FieldByName(pavKiekis)->AsInteger == table1->FieldByName(pavKiekis)->AsInteger)
//pirmi kiekiai lygus
           {

SandorisNeivyko(table1,table3,pradLaikasP,pabLaikasP,arIvykoSandoris,paskLaikasPr,paskLaikasPb);
            return;
           }
           else //pirmi kiekiai nelygus
           {
            pglSkirtumas = table1->FieldByName(pavKiekis)->AsInteger - kiekisS;
            pglLaikas = table2->FieldByName(pradLaikasP)->AsString;
            laikasPriesSand = paskLaikasPb;

SandorisIvyko(table1,table3,pradLaikasP,pabLaikasP,pglSkirtumas,pglLaikas,arIvykoSandoris,arIrasyta,par
d,paskLaikasPr,paskLaikasPb,laikasPriesSand);
           }
          }
          else //pirmos kainos nelygios
          {
           if(table2->FieldByName(pavKaina)->AsFloat > table1->FieldByName(pavKaina)->AsFloat) //kaina
padidejo
           {
            if(pard) //pardavimai
            {
             pglSkirtumas = table1->FieldByName(pavKiekis)->AsInteger - kiekisS;
             pglLaikas = table2->FieldByName(pradLaikasP)->AsString;
             laikasPriesSand = paskLaikasPb;

SandorisIvyko(table1,table3,pradLaikasP,pabLaikasP,pglSkirtumas,pglLaikas,arIvykoSandoris,arIrasyta,par
d,paskLaikasPr,paskLaikasPb,laikasPriesSand);
            }
            else //pirkimai
            {
             ArYraKaina(table2,pavKaina,pavKiekis,table1->FieldByName(pavKaina)-
>AsFloat,pglKiekis,arYra);
             if(arYra)
             {
              if(table1->FieldByName(pavKiekis)->AsInteger == pglKiekis)
              {

SandorisNeivyko(table1,table3,pradLaikasP,pabLaikasP,arIvykoSandoris,paskLaikasPr,paskLaikasPb);
               return;
              }
              else
              {
               pglSkirtumas = table1->FieldByName(pavKiekis)->AsInteger - kiekisS;
               pglLaikas = table2->FieldByName(pradLaikasP)->AsString;
               laikasPriesSand = paskLaikasPb;

SandorisIvyko(table1,table3,pradLaikasP,pabLaikasP,pglSkirtumas,pglLaikas,arIvykoSandoris,arIrasyta,par
d,paskLaikasPr,paskLaikasPb,laikasPriesSand);
              }
             }
             else
             {
              pglSkirtumas = table1->FieldByName(pavKiekis)->AsInteger - kiekisS;
              pglLaikas = table2->FieldByName(pradLaikasP)->AsString;
              laikasPriesSand = paskLaikasPb;

SandorisIvyko(table1,table3,pradLaikasP,pabLaikasP,pglSkirtumas,pglLaikas,arIvykoSandoris,arIrasyta,par
d,paskLaikasPr,paskLaikasPb,laikasPriesSand);
             }
            }
           }
           else //kaina sumazejo
           {
            if(pirk) //pirkimai
            {
             pglSkirtumas = table1->FieldByName(pavKiekis)->AsInteger - kiekisS;
             pglLaikas = table2->FieldByName(pradLaikasP)->AsString;
             laikasPriesSand = paskLaikasPb;
```

```
SandorisIvyko(table1,table3,pradLaikasP,pabLaikasP,pglSkirtumas,pglLaikas,arIvykoSandoris,arIrasyta,par
d,paskLaikasPr,paskLaikasPb,laikasPriesSand);
            }
            else //pardavimai
            {
             ArYraKaina(table2,pavKaina,pavKiekis,table1->FieldByName(pavKaina)-
>AsFloat,pglKiekis,arYra);
             if(arYra)
             {
              if(table1->FieldByName(pavKiekis)->AsInteger == pglKiekis)
              {

SandorisNeivyko(table1,table3,pradLaikasP,pabLaikasP,arIvykoSandoris,paskLaikasPr,paskLaikasPb);
               return;
              }
              else
              {
               pglSkirtumas = table1->FieldByName(pavKiekis)->AsInteger - kiekisS;
               pglLaikas = table2->FieldByName(pradLaikasP)->AsString;
               laikasPriesSand = paskLaikasPb;

SandorisIvyko(table1,table3,pradLaikasP,pabLaikasP,pglSkirtumas,pglLaikas,arIvykoSandoris,arIrasyta,par
d,paskLaikasPr,paskLaikasPb,laikasPriesSand);
              }
             }
             else
             {
              pglSkirtumas = table1->FieldByName(pavKiekis)->AsInteger - kiekisS;
              pglLaikas = table2->FieldByName(pradLaikasP)->AsString;
              laikasPriesSand = paskLaikasPb;

SandorisIvyko(table1,table3,pradLaikasP,pabLaikasP,pglSkirtumas,pglLaikas,arIvykoSandoris,arIrasyta,par
d,paskLaikasPr,paskLaikasPb,laikasPriesSand);
              }
             }
           }
          }
          }
         else // kiekis < kiekisS
         {

SandorisNeivyko(table1,table3,pradLaikasP,pabLaikasP,arIvykoSandoris,paskLaikasPr,paskLaikasPb);
          return;
         }
        }
      }
      //---------------------------------------------------------------------------
      void TDuomenuIterpimas::SandorisNeivyko(TTable*table1,TTable*table3,AnsiString pradLaikasP,
                                              AnsiString pabLaikasP,bool&arIvykoSandoris,AnsiString
&paskLaikasPr,AnsiString &paskLaikasPb)
      {
        arIvykoSandoris = false;

        if(table1->FieldByName(pradLaikasP)->AsString != paskLaikasPr
           && table1->FieldByName(pabLaikasP)->AsString != paskLaikasPb
           && table1->FieldByName(pradLaikasP)->AsString > paskLaikasPb)
         {
          PerkeltiLentele(table1,table3,"",false);
          paskLaikasPr = table1->FieldByName(pradLaikasP)->AsString;
          paskLaikasPb = table1->FieldByName(pabLaikasP)->AsString ;
         }

      }
      //---------------------------------------------------------------------------
      void TDuomenuIterpimas::SandorisIvyko(TTable *table1, TTable*table3,
                                            AnsiString pradLaikasP, AnsiString pabLaikasP,
                                            int kiekis, AnsiString nextLentLaikas,
                                            bool&arIvykoSandoris, bool &arIrasyta,bool pard,
                                            AnsiString &paskLaikasPr,AnsiString
&paskLaikasPb,AnsiString&laikasPriesSand)
      {
       AnsiString laikas, lentLaikas;
       vector <SRGLentele> naujaLentele;

       arIvykoSandoris = true;

       lentLaikas = table1->FieldByName(table1->Fields->Fields[3]->FieldName)->AsString;

       if(table1->FieldByName(pradLaikasP)->AsString != paskLaikasPr
```

```
           && table1->FieldByName(pabLaikasP)->AsString != paskLaikasPb
           && table1->FieldByName(pradLaikasP)->AsString > paskLaikasPb) //nuotrauka prieš sandorį
         {
          PerkeltiLentele(table1,table3,"",false);
          paskLaikasPr = table1->FieldByName(pradLaikasP)->AsString;
          paskLaikasPb = table1->FieldByName(pabLaikasP)->AsString;
          laikasPriesSand = paskLaikasPb;
         }

        laikas = PridetiSekundes(lentLaikas,1);
        if(!LygintiLaikus(lentLaikas.SubString(1,5),laikas.SubString(1,5)) || nextLentLaikas <= laikas)
        {
          arIrasyta = false;
          return;
        }
        else
          arIrasyta = true;


       if(kiekis == 0)
        IstrintiIrasa(table1,naujaLentele,laikas,pard);
       else
        PakeistiIrasa(table1,naujaLentele,laikas,kiekis,pard);


       IrasytiLentele(naujaLentele,table3);  //nuotrauka po sandorio
       paskLaikasPr = laikas;
       paskLaikasPb = laikas;
      }
      //---------------------------------------------------------------------------
      void TDuomenuIterpimas::Iterpti212(TTable*table1,TTable*table2,AnsiString laikasS,
                                         double kainaS,int kiekisS,bool&arIrasyta,

AnsiString&paskLaikasPr,AnsiString&paskLaikasPb,AnsiString&laikasPriesSand)
      {
        int index = 0;
        AnsiString pavKiekis,pavKaina,laikas;
        bool pard = false;
        bool pirk = false;
        vector<SRGLentele> naujaLentele;

        table1->Active = true;

        AnsiString pradLaikasP = table1->Fields->Fields[2]->FieldName;
        AnsiString pabLaikasP = table1->Fields->Fields[3]->FieldName;


        KurKaina(table1,index,pirk,pard,pavKiekis,pavKaina,kainaS);

        if(!pirk && !pard) //kainos nera nei vienam stulpely
        {

KainosNera(table1,table2,laikasS,kainaS,kiekisS,pard,pirk,arIrasyta,paskLaikasPr,paskLaikasPb,laikasPri
esSand);
         return;
        }

        if(index == 1) //kaina pirma eiluteje

NagrinetiLentele(table1,table2,pavKiekis,kainaS,kiekisS,laikasS,pard,arIrasyta,paskLaikasPr,paskLaikasP
b,laikasPriesSand);
        else //kaina ne pirma eiluteje
        {
         if(table1->FieldByName(pradLaikasP)->AsString != paskLaikasPr
           && table1->FieldByName(pabLaikasP)->AsString != paskLaikasPb
           && table1->FieldByName(pradLaikasP)->AsString > paskLaikasPb)
         {
          PerkeltiLentele(table1,table2,"",false);
          paskLaikasPr = table1->FieldByName(pradLaikasP)->AsString;
          paskLaikasPb = table1->FieldByName(pabLaikasP)->AsString;
         }


         for(int i = 1; i < index; i++)
         {
          laikas = PridetiSekundes(table1->FieldByName(pabLaikasP)->AsString,1);
          if(!LygintiLaikus(laikasS,laikas.SubString(1,5)) || laikas <= paskLaikasPb || laikas ==
paskLaikasPr)
          {
           arIrasyta = false;
```

```
          return;
         }
         else
          arIrasyta = true;

         naujaLentele.clear();
         IstrintiIrasa(table1,naujaLentele,laikas,pard);
         table1->Active = false;
         table1->EmptyTable();

         IrasytiLentele(naujaLentele,table2);
         paskLaikasPr = laikas;
         paskLaikasPb = laikas;

         IrasytiLentele(naujaLentele,table1);
         table1->Active = true;
        }


NagrinetiLentele(table1,table2,pavKiekis,kainaS,kiekisS,laikasS,pard,arIrasyta,paskLaikasPr,paskLaikasP
b,laikasPriesSand);
       }
      }
      //---------------------------------------------------------------------------
      void TDuomenuIterpimas::KainosNera(TTable*table1,TTable*table2,AnsiString laikasS,double kainaS,
                                         int kiekisS, bool &pard,bool &pirk,bool &arIrasyta,

AnsiString&paskLaikasPr,AnsiString&paskLaikasPb,AnsiString&laikasPriesSand)
      {
        AnsiString laikas;
        vector<SRGLentele> naujaLentele;

        AnsiString pradLaikasP = table1->Fields->Fields[2]->FieldName;
        AnsiString pabLaikasP = table1->Fields->Fields[3]->FieldName;

        AnsiString pardKainaP = table1->Fields->Fields[7]->FieldName;
        AnsiString pirkKainaP = table1->Fields->Fields[5]->FieldName;

        double skirtumas1 = table1->FieldByName(pardKainaP)->AsFloat - kainaS;
        double skirtumas2 = kainaS - table1->FieldByName(pirkKainaP)->AsFloat;
         if(skirtumas1 < 0 || skirtumas2 < 0)
         {
          arIrasyta = false;
          return;
         }

         double eps = 1e-8;

         if(skirtumas1 + eps < skirtumas2)
           pard = true;
         else
          if (skirtumas1 > skirtumas2 + eps)
           pirk = true;
          else
          {
           double sk = rand()*1.0/RAND_MAX;
           if(sk <= 0.5)
            pard = true;
           else
            pirk = true;
          }


         if(table1->FieldByName(pradLaikasP)->AsString != paskLaikasPr
            && table1->FieldByName(pabLaikasP)->AsString != paskLaikasPb
            && table1->FieldByName(pradLaikasP)->AsString > paskLaikasPb)
         {
           PerkeltiLentele(table1,table2,"",false);
           paskLaikasPr = table1->FieldByName(pradLaikasP)->AsString;
           paskLaikasPb = table1->FieldByName(pabLaikasP)->AsString;
         }

         laikas = PridetiSekundes(table1->FieldByName(pabLaikasP)->AsString,1);
         if(!LygintiLaikus(laikasS,laikas.SubString(1,5)) || laikas <= paskLaikasPb || laikas ==
paskLaikasPr)
         {
          arIrasyta = false;
          return;
         }
         else
```

```
          arIrasyta = true;


         IterptiIrasa(table1,naujaLentele,laikas,kainaS,kiekisS,pard);
         IrasytiLentele(naujaLentele,table2);  //nuotrauka pries sandori
         paskLaikasPr = laikas;
         paskLaikasPb = laikas;
         laikasPriesSand = paskLaikasPb;

         laikas = PridetiSekundes(table1->FieldByName(table1->Fields->Fields[3]->FieldName)-
>AsString,2);
         if(!LygintiLaikus(laikasS,laikas.SubString(1,5)) || laikas <= paskLaikasPb || laikas ==
paskLaikasPr)
         {
          arIrasyta = false;
          return;
         }
         else
          arIrasyta = true;

          PerkeltiLentele(table1,table2,laikas,true);  //nuotrauka po sandorio
          paskLaikasPr = laikas;
          paskLaikasPb = laikas;
      }
      //---------------------------------------------------------------------------
      bool TDuomenuIterpimas::LygintiLaikus(AnsiString laikas1, AnsiString laikas2)
      {
        int min1 = StrToInt(laikas1.SubString(1,2))*60 + StrToInt(laikas1.SubString(4,2));
        int min2 = StrToInt(laikas2.SubString(1,2))*60 + StrToInt(laikas2.SubString(4,2));

        if(min1 == min2)
         return true;
        else
         return false;

      }
      //---------------------------------------------------------------------------
      void TDuomenuIterpimas::NagrinetiLentele(TTable *table1, TTable *table2,
                                               AnsiString pavKiekis,
                                               double kainaS, int kiekisS,AnsiString laikasS,bool pard,
                                               bool
&arIrasyta,AnsiString&paskLaikasPr,AnsiString&paskLaikasPb,
                                               AnsiString &laikasPriesSand)
      {
        vector <SRGLentele> naujaLentele;
        int itKiekis;
        AnsiString laikas;
        AnsiString pradLaikasP = table1->Fields->Fields[2]->FieldName;
        AnsiString pabLaikasP = table1->Fields->Fields[3]->FieldName;

        if(table1->FieldByName(pradLaikasP)->AsString != paskLaikasPr
            && table1->FieldByName(pabLaikasP)->AsString != paskLaikasPb
            && table1->FieldByName(pradLaikasP)->AsString > paskLaikasPb)
         {
           PerkeltiLentele(table1,table2,"",false);
           paskLaikasPr = table1->FieldByName(pradLaikasP)->AsString;
           paskLaikasPb = table1->FieldByName(pabLaikasP)->AsString;

         }
        laikasPriesSand = paskLaikasPb;

        laikas = PridetiSekundes(table1->FieldByName(pabLaikasP)->AsString,1);
        if(!LygintiLaikus(laikasS,laikas.SubString(1,5)) || laikas <= paskLaikasPb || laikas ==
paskLaikasPr)
        {
         arIrasyta = false;
         return;
        }
        else
        arIrasyta = true;

        if(kiekisS < table1->FieldByName(pavKiekis)->AsInteger)
        {
         itKiekis = table1->FieldByName(pavKiekis)->AsInteger - kiekisS;
         PakeistiIrasa(table1,naujaLentele,laikas,itKiekis,pard);

         IrasytiLentele(naujaLentele,table2);  //nuotrauka po sandorio
         paskLaikasPr = laikas;
         paskLaikasPb = laikas;
        }
```

```
        else
         if(kiekisS == table1->FieldByName(pavKiekis)->AsInteger)
         {
           IstrintiIrasa(table1,naujaLentele,laikas,pard);
           IrasytiLentele(naujaLentele,table2);  //nuotrauka po sandorio
           paskLaikasPr = laikas;
           paskLaikasPb = laikas;
         }
         else
         {
          IterptiIrasa(table1,naujaLentele,laikas,kainaS,kiekisS,pard);
          IrasytiLentele(naujaLentele,table2);  //nuotrauka pries sandori
          paskLaikasPr = laikas;
          paskLaikasPb = laikas;
          laikasPriesSand = paskLaikasPb;

          laikas = PridetiSekundes(table1->FieldByName(pabLaikasP)->AsString,2); //nuotrauka po
sandorio
          if(!LygintiLaikus(laikasS,laikas.SubString(1,5)) || laikas <= paskLaikasPb || laikas ==
paskLaikasPr)
          {
           arIrasyta = false;
           return;
          }
          else
           arIrasyta = true;


          PerkeltiLentele(table1,table2,laikas,true);  //nuotrauka po sandorio
          paskLaikasPr = laikas;
          paskLaikasPb = laikas;
         }
      }
      //---------------------------------------------------------------------------
      AnsiString TDuomenuIterpimas::PridetiSekundes(AnsiString laikas,int sek)
      {
        int pglSek;
        div_t x,y;
        AnsiString val,min,sekk;

        pglSek = StrToInt(laikas.SubString(1,2))*3600
               + StrToInt(laikas.SubString(4,2))*60
               + StrToInt(laikas.SubString(7,2)) + sek;

        x = div(pglSek,3600);
        y = div(x.rem,60);

        if(x.quot < 10)
         val = "0" + IntToStr(x.quot);
        else
         val = IntToStr(x.quot);

        if(y.quot < 10)
         min = "0" + IntToStr(y.quot);
        else
         min = IntToStr(y.quot);

        if(y.rem < 10)
         sekk = "0" + IntToStr(y.rem);
        else
         sekk = IntToStr(y.rem);


        AnsiString nLaikas = val + ":" + min + ":" + sekk;

        return nLaikas;

      }
      //---------------------------------------------------------------------------
      void TDuomenuIterpimas::IstrintiIrasa(TTable*lentele,vector <SRGLentele> &naujaLentele,
                                            AnsiString laikas, bool pard)
      {
        AnsiString dataP = lentele->Fields->Fields[0]->FieldName;
        AnsiString akcijaP = lentele->Fields->Fields[1]->FieldName;
        AnsiString pirkKiekisP = lentele->Fields->Fields[4]->FieldName;
        AnsiString pirkKainaP = lentele->Fields->Fields[5]->FieldName;
        AnsiString valiutaPi = lentele->Fields->Fields[6]->FieldName;
        AnsiString pardKainaP = lentele->Fields->Fields[7]->FieldName;
        AnsiString valiutaPa = lentele->Fields->Fields[8]->FieldName;
        AnsiString pardKiekisP = lentele->Fields->Fields[9]->FieldName;
```

```
vector<SRGStulpelis> pglVektorius;

double pirkKaina,pardKaina;
int pirkKiekis,pardKiekis,ilgis;


if(pard)
{
  lentele->First();
  while(!lentele->Eof)
  {
   pirkKaina = lentele->FieldByName(pirkKainaP)->AsFloat;
   pirkKiekis = lentele->FieldByName(pirkKiekisP)->AsInteger;
   pardKaina = lentele->FieldByName(pardKainaP)->AsFloat;
   pardKiekis = lentele->FieldByName(pardKiekisP)->AsInteger;

   if(pirkKiekis != 0 && pirkKaina != 0)
     naujaLentele.push_back(SRGLentele(lentele->FieldByName(dataP)->AsString,
                                       lentele->FieldByName(akcijaP)->AsString,
                                       laikas,laikas,pirkKiekis,pirkKaina,
                                       lentele->FieldByName(valiutaPi)->AsString,
                                       0,"",0));
   if(pardKaina != 0 && pardKiekis != 0)
     pglVektorius.push_back(SRGStulpelis(lentele->FieldByName(pardKainaP)->AsFloat,
                                         lentele->FieldByName(valiutaPa)->AsString,
                                         lentele->FieldByName(pardKiekisP)->AsInteger));

   lentele->Next();
  }
  ilgis = pglVektorius.size() - 1;
  if(ilgis <= (int)naujaLentele.size())
   for(int i = 1; i < (int)pglVektorius.size(); i++)
   {
    naujaLentele[i - 1].pardKaina = pglVektorius[i].kaina;
    naujaLentele[i - 1].pardKiekis = pglVektorius[i].kiekis;
    naujaLentele[i - 1].pardVal = pglVektorius[i].valiuta;
   }
  else
  {
   for(int i = 1; i <= (int)naujaLentele.size(); i++)
   {
    naujaLentele[i - 1].pardKaina = pglVektorius[i].kaina;
    naujaLentele[i - 1].pardKiekis = pglVektorius[i].kiekis;
    naujaLentele[i - 1].pardVal = pglVektorius[i].valiuta;
   }
   for(int i = naujaLentele.size() + 1; i < (int) pglVektorius.size(); i++)
    naujaLentele.push_back(SRGLentele(lentele->FieldByName(dataP)->AsString,
                                      lentele->FieldByName(akcijaP)->AsString,
                                      laikas,laikas,0,0,"",
                                      pglVektorius[i].kaina,
                                      lentele->FieldByName(valiutaPa)->AsString,
                                      pglVektorius[i].kiekis));
  }
}
else
{
 lentele->First();
 while(!lentele->Eof)
 {
  pirkKaina = lentele->FieldByName(pirkKainaP)->AsFloat;
  pirkKiekis = lentele->FieldByName(pirkKiekisP)->AsInteger;
  pardKaina = lentele->FieldByName(pardKainaP)->AsFloat;
  pardKiekis = lentele->FieldByName(pardKiekisP)->AsInteger;

  if(pardKaina != 0 && pardKiekis != 0)
    naujaLentele.push_back(SRGLentele(lentele->FieldByName(dataP)->AsString,
                                      lentele->FieldByName(akcijaP)->AsString,
                                      laikas,laikas,0,0,"",
                                      lentele->FieldByName(pardKainaP)->AsFloat,
                                      lentele->FieldByName(valiutaPa)->AsString,
                                      lentele->FieldByName(pardKiekisP)->AsInteger));


   if(pirkKiekis != 0 && pirkKaina != 0)
    pglVektorius.push_back(SRGStulpelis(lentele->FieldByName(pirkKainaP)->AsFloat,
                                        lentele->FieldByName(valiutaPi)->AsString,
                                        lentele->FieldByName(pirkKiekisP)->AsInteger));

   lentele->Next();
```

```
    }
    ilgis = pglVektorius.size() - 1;

    if(ilgis <= (int) naujaLentele.size())
     for(int i = 1; i < (int)pglVektorius.size(); i++)
     {
      naujaLentele[i - 1].pirkKaina = pglVektorius[i].kaina;
      naujaLentele[i - 1].pirkKiekis = pglVektorius[i].kiekis;
      naujaLentele[i - 1].pirkVal = pglVektorius[i].valiuta;
     }
    else
    {
     for(int i = 1; i <= (int)naujaLentele.size(); i++)
     {
      naujaLentele[i - 1].pirkKaina = pglVektorius[i].kaina;
      naujaLentele[i - 1].pirkKiekis = pglVektorius[i].kiekis;
      naujaLentele[i - 1].pirkVal = pglVektorius[i].valiuta;
     }
     for(int i = naujaLentele.size() + 1; i < (int) pglVektorius.size(); i++)
      naujaLentele.push_back(SRGLentele(lentele->FieldByName(dataP)->AsString,
                                        lentele->FieldByName(akcijaP)->AsString,
                                        laikas,laikas,
                                        pglVektorius[i].kiekis,
                                        pglVektorius[i].kaina,
                                        lentele->FieldByName(valiutaPi)->AsString,
                                        0,"",0));
    }
  }
  lentele->First();
}
//---------------------------------------------------------------------------
void TDuomenuIterpimas::IterptiIrasa(TTable*lentele,vector <SRGLentele> &naujaLentele,
                                     AnsiString laikas,double kainaI,int kiekisI, bool pard)
{
  AnsiString dataP = lentele->Fields->Fields[0]->FieldName;
  AnsiString akcijaP = lentele->Fields->Fields[1]->FieldName;
  AnsiString pirkKiekisP = lentele->Fields->Fields[4]->FieldName;
  AnsiString pirkKainaP = lentele->Fields->Fields[5]->FieldName;
  AnsiString valiutaPi = lentele->Fields->Fields[6]->FieldName;
  AnsiString pardKainaP = lentele->Fields->Fields[7]->FieldName;
  AnsiString valiutaPa = lentele->Fields->Fields[8]->FieldName;
  AnsiString pardKiekisP = lentele->Fields->Fields[9]->FieldName;

  vector<SRGStulpelis> pglVektorius;

  double pirkKaina,pardKaina;
  int pirkKiekis,pardKiekis,ilgis;


  if(pard)
  {
    lentele->First();
    while(!lentele->Eof)
    {
     pirkKaina = lentele->FieldByName(pirkKainaP)->AsFloat;
     pirkKiekis = lentele->FieldByName(pirkKiekisP)->AsInteger;
     pardKaina = lentele->FieldByName(pardKainaP)->AsFloat;
     pardKiekis = lentele->FieldByName(pardKiekisP)->AsInteger;

     if(pirkKiekis != 0 && pirkKaina != 0)
       naujaLentele.push_back(SRGLentele(lentele->FieldByName(dataP)->AsString,
                                         lentele->FieldByName(akcijaP)->AsString,
                                         laikas,laikas,pirkKiekis,pirkKaina,
                                         lentele->FieldByName(valiutaPi)->AsString,
                                         0,"",0));
     if(pardKaina != 0 && pardKiekis != 0)
       pglVektorius.push_back(SRGStulpelis(lentele->FieldByName(pardKainaP)->AsFloat,
                                           lentele->FieldByName(valiutaPa)->AsString,
                                           lentele->FieldByName(pardKiekisP)->AsInteger));

     lentele->Next();
    }

    ilgis = pglVektorius.size() + 1;
    if(ilgis <= (int) naujaLentele.size())
     for(int i = 0; i <= (int)pglVektorius.size(); i++)
     {
      if(i == 0)
      {
       naujaLentele[i].pardKaina = kainaI;
```

```
      naujaLentele[i].pardKiekis = kiekisI;
      naujaLentele[i].pardVal = naujaLentele[i].pirkVal;
     }
     else
     {
      naujaLentele[i].pardKaina = pglVektorius[i - 1].kaina;
      naujaLentele[i].pardKiekis = pglVektorius[i - 1].kiekis;
      naujaLentele[i].pardVal = pglVektorius[i - 1].valiuta;
     }
    }
   else
   {
    for(int i = 0; i < (int)naujaLentele.size(); i++)
    {
     if(i == 0)
     {
      naujaLentele[i].pardKaina = kainaI;
      naujaLentele[i].pardKiekis = kiekisI;
      naujaLentele[i].pardVal = naujaLentele[i].pirkVal;
     }
     else
     {
      naujaLentele[i].pardKaina = pglVektorius[i - 1].kaina;
      naujaLentele[i].pardKiekis = pglVektorius[i - 1].kiekis;
      naujaLentele[i].pardVal = pglVektorius[i - 1].valiuta;
     }
    }
    for(int i = naujaLentele.size() - 1; i < (int) pglVektorius.size(); i++)
     naujaLentele.push_back(SRGLentele(lentele->FieldByName(dataP)->AsString,
                                       lentele->FieldByName(akcijaP)->AsString,
                                       laikas,laikas,0,0,"",
                                       pglVektorius[i].kaina,
                                       lentele->FieldByName(valiutaPa)->AsString,
                                       pglVektorius[i].kiekis));
   }
}
else
{
 lentele->First();
 while(!lentele->Eof)
 {
  pirkKaina = lentele->FieldByName(pirkKainaP)->AsFloat;
  pirkKiekis = lentele->FieldByName(pirkKiekisP)->AsInteger;
  pardKaina = lentele->FieldByName(pardKainaP)->AsFloat;
  pardKiekis = lentele->FieldByName(pardKiekisP)->AsInteger;

  if(pardKaina != 0 && pardKiekis != 0)
    naujaLentele.push_back(SRGLentele(lentele->FieldByName(dataP)->AsString,
                                      lentele->FieldByName(akcijaP)->AsString,
                                      laikas,laikas,0,0,"",
                                      lentele->FieldByName(pardKainaP)->AsFloat,
                                      lentele->FieldByName(valiutaPa)->AsString,
                                      lentele->FieldByName(pardKiekisP)->AsInteger));


   if(pirkKiekis != 0 && pirkKaina != 0)
    pglVektorius.push_back(SRGStulpelis(lentele->FieldByName(pirkKainaP)->AsFloat,
                                        lentele->FieldByName(valiutaPi)->AsString,
                                        lentele->FieldByName(pirkKiekisP)->AsInteger));

   lentele->Next();
  }
  ilgis = pglVektorius.size() + 1;

  if(ilgis <= (int) naujaLentele.size())
   for(int i = 0; i <= (int)pglVektorius.size(); i++)
   {
    if(i == 0)
    {
     naujaLentele[i].pirkKaina = kainaI;
     naujaLentele[i].pirkKiekis = kiekisI;
     naujaLentele[i].pirkVal = naujaLentele[i].pardVal;
    }
    else
    {
     naujaLentele[i].pirkKaina = pglVektorius[i - 1].kaina;
     naujaLentele[i].pirkKiekis = pglVektorius[i - 1].kiekis;
     naujaLentele[i].pirkVal = pglVektorius[i - 1].valiuta;
    }
   }
```

```
    else
    {
     for(int i = 0; i < (int)naujaLentele.size(); i++)
     {
      if(i == 0)
      {
       naujaLentele[i].pirkKaina = kainaI;
       naujaLentele[i].pirkKiekis = kiekisI;
       naujaLentele[i].pirkVal = naujaLentele[i].pardVal;
      }
      else
      {
       naujaLentele[i].pirkKaina = pglVektorius[i - 1].kaina;
       naujaLentele[i].pirkKiekis = pglVektorius[i - 1].kiekis;
       naujaLentele[i].pirkVal = pglVektorius[i - 1].valiuta;
      }
     }
     for(int i = naujaLentele.size() - 1; i < (int) pglVektorius.size(); i++)
      naujaLentele.push_back(SRGLentele(lentele->FieldByName(dataP)->AsString,
                                        lentele->FieldByName(akcijaP)->AsString,
                                        laikas,laikas,
                                        pglVektorius[i].kiekis,
                                        pglVektorius[i].kaina,
                                        lentele->FieldByName(valiutaPi)->AsString,
                                        0,"",0));
    }
  }
  lentele->First();
}
//---------------------------------------------------------------------------
void TDuomenuIterpimas::PakeistiIrasa(TTable*lentele,vector <SRGLentele> &naujaLentele,
                                      AnsiString laikas,int kiekisI, bool pard)
{
  AnsiString dataP = lentele->Fields->Fields[0]->FieldName;
  AnsiString akcijaP = lentele->Fields->Fields[1]->FieldName;
  AnsiString pirkKiekP = lentele->Fields->Fields[4]->FieldName;
  AnsiString pirkKainaP = lentele->Fields->Fields[5]->FieldName;
  AnsiString valiutaPi = lentele->Fields->Fields[6]->FieldName;
  AnsiString pardKainaP = lentele->Fields->Fields[7]->FieldName;
  AnsiString valiutaPa = lentele->Fields->Fields[8]->FieldName;
  AnsiString pardKiekisP = lentele->Fields->Fields[9]->FieldName;


  lentele->First();

  while(!lentele->Eof)
  {
   naujaLentele.push_back(SRGLentele(lentele->FieldByName(dataP)->AsString,
                                     lentele->FieldByName(akcijaP)->AsString,
                                     laikas,laikas,
                                     lentele->FieldByName(pirkKiekP)->AsInteger,
                                     lentele->FieldByName(pirkKainaP)->AsFloat,
                                     lentele->FieldByName(valiutaPi)->AsString,
                                     lentele->FieldByName(pardKainaP)->AsFloat,
                                     lentele->FieldByName(valiutaPa)->AsString,
                                     lentele->FieldByName(pardKiekisP)->AsInteger));
   lentele->Next();
  }

  if(pard)
   naujaLentele[0].pardKiekis = kiekisI;
  else
   naujaLentele[0].pirkKiekis = kiekisI;

 lentele->First();
}
//---------------------------------------------------------------------------
void TDuomenuIterpimas::LygintiIrasus(TQuery*uzklausa,TTable*table1,TTable*table2,TTable*table3)
{
 AnsiString uzTekstas,pgl;
 AnsiString pradLaikasP,pabLaikasP;
 SLaikai *sarLaikas;
 int sarLaikasIlgis;

 table1->Active = false;
 table2->Active = false;

 AnsiString LentelesPav1 = table1->TableName;
 AnsiString DBVardas1 = table1->DatabaseName;
```

```
 table2->DatabaseName = DBVardas1;
 table2->TableName = LentelesPav1;

 table1->Active = true;

 pradLaikasP = table1->Fields->Fields[2]->FieldName;
 pabLaikasP = table1->Fields->Fields[3]->FieldName;

  uzTekstas = "select distinct " + pradLaikasP +", " + pabLaikasP + " from " + LentelesPav1
              + "' order by " + pradLaikasP;

  DuomSut.SudarytiLaikuSarasa(uzklausa,uzTekstas,pradLaikasP,pabLaikasP,sarLaikas,
                      sarLaikasIlgis,DBVardas1);

  for(int k = 0; k < sarLaikasIlgis - 1; k ++)
  {
   pgl = pradLaikasP + " = '" + sarLaikas[k].laikasSek + "'";
   FiltruotiLentele(table1,pgl);

   for(int kk = k + 1; kk < sarLaikasIlgis; kk ++)
   {
    pgl = pradLaikasP + " = '" + sarLaikas[kk].laikasSek + "'";
    FiltruotiLentele(table2,pgl);
    table3->Active = true;

    if(!DuomSut.ArLentelesVienodos(table1,table2))
    {
     DuomSut.IrasytiLentele(table1,table3,sarLaikas[k].laikasSek,sarLaikas[kk - 1].laikas);
     k = kk - 1;
     kk = sarLaikasIlgis;
     if(k + 1 == sarLaikasIlgis - 1)
      DuomSut.IrasytiLentele(table2,table3,sarLaikas[k + 1].laikasSek,sarLaikas[k + 1].laikas);
    }
    else if(kk == sarLaikasIlgis - 1)
    {
     DuomSut.IrasytiLentele(table1,table3,sarLaikas[k].laikas,sarLaikas[kk].laikas);
     k = kk - 1;
    }
   }
  }

  table1->Active = false;
  table2->Active = false;
  table1->EmptyTable();
  table2->EmptyTable();


  delete [] sarLaikas;
}
//---------------------------------------------------------------------------
```

„DB_Analize“

„Pagrindinis.h“

```
//---------------------------------------------------------------------------
class TForm1 : public TForm
{
private:             // User declarations

        AnsiString kelias; //kelias iki programos katalogo
        AnsiString keliasP; //kelias iki pagrindinės lentelės duomenų katalogo
        AnsiString keliasS; //kelias iki sandorių duomenų katalogo
        AnsiString keliasG; //kelias iki rinkos gylio duomenų katalogo
        AnsiString lenteleP; //pagrindinė lentelė
        AnsiString lenteleS; //sandorių lentelė
        AnsiString lenteleG; //rinkos gylio lentelė

        bool arAkumuliuotas;
        bool arRGylis;
        bool arKainuPokytis;
        bool arSandoriai;


         vector <double> eSpreadV;
         vector <SSand> sandoriaiV;
         vector <double> qSpreadV;

public:              // User declarations
```

```
        TBendraAnalize BendraA; //klasės "TBendraAnalize" objektas
        TStacionarumas Stac; //klasės "TStacionarumas" objektas
        TSpread Spread; //klasės "TSpread" objektas
        TEfektyvumas Efekt; ////klasės "TEfektyvumas" objektas
         __fastcall TForm1(TComponent* Owner);
};
```

Pagrindinis.cpp

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Pagrindinis.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
  kelias = GetCurrentDir();
  keliasP = GetCurrentDir() +  "\\Duomenys";
  keliasS = GetCurrentDir() +  "\\Duomenys\\Sandoriai";
  keliasG = GetCurrentDir() +  "\\Duomenys\\RGylis";
  lenteleP  = "TPagrindine.dbf";
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Analiz1Click(TObject *Sender)
{
  this->TabSheet2->TabVisible = true;
  this->PageControl1->Pages[0]->Show();
  this->Memo1->Clear();
  this->ComboBox1->Clear();
  this->ComboBox2->Clear();
  this->ComboBox3->Clear();

 AnsiString kriterijus = "Trumpinys";
 AnsiString  uzTekstas = "select distinct " + kriterijus + " from " + lenteleP +  " order by " +
kriterijus;
 BendraA.UzpildytiComboBox(this->Query1,this->ComboBox1,keliasP,lenteleP,kriterijus,uzTekstas);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::ComboBox1Click(TObject *Sender)
{

  AnsiString akcija = this->ComboBox1->Text;
  lenteleS = "S" + akcija + ".dbf";
  lenteleG = "G" + akcija + ".dbf";;
  BendraA.UzpildytiMemo(this->Memo1,this-
>Query1,keliasP,keliasS,keliasG,lenteleP,lenteleS,lenteleG,akcija);


 AnsiString kriterijus = "Data";
 AnsiString kriterijus1 = "Trumpinys";

 AnsiString uzTekstas = "select distinct " + kriterijus + " from " + lenteleP
                      +  " where " + kriterijus1 + " = '" + akcija  + "' order by " + kriterijus;
 BendraA.UzpildytiComboBox(this->Query1,this->ComboBox2,keliasP,lenteleP,kriterijus,uzTekstas);
 BendraA.UzpildytiComboBox(this->Query1,this->ComboBox3,keliasP,lenteleP,kriterijus,uzTekstas);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
 AnsiString data,akcija;
 vector <SSkirtumai> sandKainos;
 vector <SSkirtumai> skirtumaiV;
 vector <double> meanX;
 vector <double> meanX2;
 vector <double> meanXpgl;
 vector <double> meanX2pgl;

 if(this->ComboBox1->ItemIndex == -1 || this->ComboBox2->ItemIndex == -1
    || this->ComboBox3->ItemIndex == -1)
 {
  ShowMessage("Pasirinkite visus duomenis!");
  return;
```

```
       }
       if(this->ComboBox2->Items->Strings[this->ComboBox2->ItemIndex] > this->ComboBox3->Items-
>Strings[this->ComboBox3->ItemIndex])
       {
        ShowMessage("Blogai pasirinkote datas");
        return;
       }

       try{
        this->TabSheet3->TabVisible = true;
        this->PageControl1->Pages[1]->Show();
        akcija = this->ComboBox1->Items->Strings[this->ComboBox1->ItemIndex];
        AnsiString pavadinimas = akcija + " akcijos įvykusių sandorių kainų skirtumų grafikas";
        this->Chart1->Title->Text->SetText(pavadinimas.c_str());
        for(int i = this->ComboBox2->ItemIndex; i <= this->ComboBox3->ItemIndex; i++)
        {
         data = this->ComboBox2->Items->Strings[i];
         AnsiString uzTekstas = "select Kaina from " + lenteleS + " where Data = '" + data + "'";
         Stac.DuomenuVektorius(this->Query1,sandKainos,keliasS,lenteleS,data,uzTekstas);
        }

        Stac.SkaiciuotiVidurkius(sandKainos,meanXpgl,meanX2pgl,skirtumaiV);
        Stac.VaizduotiDuomenis(this->Series1,skirtumaiV);
        if(this->CheckBox1->Checked)
        {
         for(int i = 0; i < (int) meanXpgl.size(); i++)
          if(div(i,2).rem == 0)
          {
            meanX.push_back(meanXpgl[i]);
            meanX2.push_back(meanX2pgl[i]);
          }
        }
        else
        {
         meanX = meanXpgl;
         meanX2 = meanX2pgl;
        }

        Stac.GridoAntraste(this->StringGrid1);
        AnsiString data1 = this->ComboBox2->Items->Strings[this->ComboBox2->ItemIndex];
        AnsiString data2 = this->ComboBox3->Items->Strings[this->ComboBox3->ItemIndex];

        Stac.RAT(this->StringGrid1,meanX,akcija,data1,data2,"EX");
        Stac.RAT(this->StringGrid1,meanX2,akcija,data1,data2,"E(X^2)");

        this->Series2->Clear();
        this->Series3->Clear();
        Stac.VaizduotiVidurkius(this->Series2,meanX);
        Stac.VaizduotiVidurkius(this->Series3,meanX2);
       }
       catch(EConvertError& ex)
       {
        ShowMessage("Blogai įvedėte intervalų skaičių!");
       }
      }
      //---------------------------------------------------------------------------
      void __fastcall TForm1::Button2Click(TObject *Sender)
      {

       vector <SRGylis> gylis;
       vector <double> qSpread;
       qSpreadV.clear();
       vector <double> eSpread;
       eSpreadV.clear();
       vector <double> uSpread;
       vector <double> uSpreadV;
       vector<SSand> RollIverciai;
       vector <SSand> sandoriai;
       sandoriaiV.clear();
          bool arViskasGerai;
       AnsiString uzTekstas,data;

       if(this->ComboBox1->ItemIndex == -1 || this->ComboBox2->ItemIndex == -1
          || this->ComboBox3->ItemIndex == -1)
       {
        ShowMessage("Pasirinkite visus duomenis!");
        return;
       }
       if(this->ComboBox2->Items->Strings[this->ComboBox2->ItemIndex] > this->ComboBox3->Items-
>Strings[this->ComboBox3->ItemIndex])
```

```
{
 ShowMessage("Blogai pasirinkote datas");
 return;
}


 this->TabSheet7->TabVisible = true;
 this->PageControl1->Pages[3]->Show();
 AnsiString akcija = this->ComboBox1->Items->Strings[this->ComboBox1->ItemIndex];


for(int i = this->ComboBox2->ItemIndex; i <= this->ComboBox3->ItemIndex; i++)
{
 gylis.clear();
 qSpread.clear();
 eSpread.clear();
 uSpread.clear();
 sandoriai.clear();

 if(this->CheckBox2->Checked || this->CheckBox3->Checked)
 {
  data = this->ComboBox2->Items->Strings[i];
  uzTekstas = "select Data,Pr_laikas,Pb_laikas,Pirk_kiek,Pirk_kaina,Pard_kaina,Pard_kiek from "
                    + lenteleG + " where Data = '" + data + "'";
  Spread.IrasytiGyli(this->Query1,gylis,keliasG,uzTekstas);
 }

 if(gylis.size() != 0) //jei prekyba nesustabdyta
 {
  //"quoted spread"
  if(this->CheckBox2->Checked)
  {
   Spread.PaprastasSpredas(gylis,qSpread,qSpreadV);
   for(int ii = 0; ii < (int) qSpread.size(); ii++)
    qSpreadV.push_back(qSpread[ii]);
  }

  if(this->CheckBox3->Checked || this->CheckBox4->Checked)
  {
   uzTekstas = "select Laikass, Kaina from "
               + lenteleS + " where Data = '" + data + "' and Laikass <> ''";

   bool arViskasGerai;
   Spread.IrasytiSandorius(this->Query1,sandoriai,keliasS,uzTekstas,arViskasGerai);
   if(!arViskasGerai)
    break;
   for(int ii = 0; ii < (int) sandoriai.size(); ii++)
    sandoriaiV.push_back(sandoriai[ii]);

   //"effective spread"
   if(this->CheckBox3->Checked)
   {
    Spread.EfektyvusSpredas(gylis,sandoriai,eSpread,eSpreadV);
    for(int ii = 0; ii < (int) eSpread.size(); ii++)
     eSpreadV.push_back(eSpread[ii]);
   }
  }
 }
}

//"užfiksuotas spredas"
 if(this->CheckBox4->Checked)
  Spread.UzfiksuotasSpredas(sandoriaiV,uSpreadV);


double ivertis;
bool arNeigiamas = true;

if(this->CheckBox5->Checked)
{
 vector <SSand> sandoriaiRoll;
 vector <SSand> sandoriaiVRoll;
 vector <double> uSpreadRoll;
 vector <double> uSpreadVRoll;

 for(int i = this->ComboBox2->ItemIndex; i <= this->ComboBox3->ItemIndex; i++)
 {
  uSpreadRoll.clear();
  sandoriaiRoll.clear();
  data = this->ComboBox2->Items->Strings[i];
```

```
  if(this->RadioButton6->Checked)
  {

    uzTekstas = "select Laikas, Kaina from "
                  + lenteleS + " where Data = '" + data + "' and Laikas <> '-'";
    Spread.IrasytiSandorius(this->Query1,sandoriaiRoll,keliasS,uzTekstas,arViskasGerai);
    if(!arViskasGerai)
     break;
  }
  else
   if (this->RadioButton7->Checked)
   {
    uzTekstas = "select Data, Uzd from "
                  + lenteleP + " where Data = '" + data + "' and Trumpinys = '" + akcija + "'";
    Spread.IrasytiSandorius(this->Query1,sandoriaiRoll,keliasP,uzTekstas,arViskasGerai);
    if(!arViskasGerai)
     break;
  }
  for(int ii = 0; ii < (int) sandoriaiRoll.size(); ii++)
   sandoriaiVRoll.push_back(sandoriaiRoll[ii]);

  Spread.UzfiksuotasSpredas(sandoriaiRoll,uSpreadRoll);
  Spread.RollSpredas(uSpreadRoll,ivertis,arNeigiamas);
  if(!arNeigiamas)
   RollIverciai.push_back(SSand(data,ivertis));
  else
   RollIverciai.push_back(SSand(data,-0.01));
 }
 Spread.UzfiksuotasSpredas(sandoriaiVRoll,uSpreadVRoll);
 Spread.RollSpredas(uSpreadVRoll,ivertis,arNeigiamas);
}

AnsiString data1 =  this->ComboBox2->Text;
AnsiString data2 = this->ComboBox3->Text;

Spread.IsvestiRezultatus(this->StringGrid2,this->CheckBox2,this->CheckBox3,
                         this->CheckBox4,this->CheckBox5,qSpreadV,eSpreadV,
                         uSpreadV,ivertis,arNeigiamas,akcija,data1,data2);


 if(this->CheckBox9->Checked)
 {
  vector <double> skirtumaiV;
  vector <SSand> pglSand;
  vector <SSand> pglSandV;
  vector <double> rollV;

  for(int i = this->ComboBox2->ItemIndex; i <= this->ComboBox3->ItemIndex; i++)
  {
   pglSand.clear();
   skirtumaiV.clear();
   data = this->ComboBox2->Items->Strings[i];
   if(this->RadioButton6->Checked)
   {
    uzTekstas = "select Laikas, Kaina from "
                  + lenteleS + " where Data = '" + data + "' and Laikas <> '-'";
    Spread.IrasytiSandorius(this->Query1,pglSand,keliasS,uzTekstas,arViskasGerai);
     if(!arViskasGerai)
      break;
   }
   else
    if (this->RadioButton7->Checked)
    {
     uzTekstas = "select Data, Uzd from "
                  + lenteleP + " where Data = '" + data + "' and Trumpinys = '" + akcija + "'";
     Spread.IrasytiSandorius(this->Query1,pglSand,keliasP,uzTekstas,arViskasGerai);
     if(!arViskasGerai)
      break;
    }
   for(int ii = 0; ii < (int) pglSand.size(); ii++)
    pglSandV.push_back(pglSand[ii]);

   Spread.UzfiksuotasSpredas(pglSandV,skirtumaiV);
   Spread.RollSpredas(skirtumaiV,ivertis,arNeigiamas);
   if(!arNeigiamas)
     rollV.push_back(ivertis);
   else
    rollV.push_back(-0.01);
  }
```

```
         this->Chart7->BottomAxis->Title->Caption = "Dienų skaičius";
         this->Chart7->Title->Text->SetText("");
         AnsiString pavadinimas = "\"Roll\" matų kitimo kreivė (sumuojant dienas) - " + akcija;
         this->Chart7->Title->Text->SetText(pavadinimas.c_str());
         Stac.VaizduotiVidurkius(this->Series9,rollV);
        }
        else
         if(this->CheckBox10->Checked)
         {
          Spread.VaizduotiIvercius(this->Series9,RollIverciai);
          this->Chart7->BottomAxis->Title->Caption = "";
          AnsiString pavadinimas = "\"Roll\" matų kitimo kreivė (kiekvienai dienai atskirai) - " +
akcija;
          this->Chart7->Title->Text->SetText(pavadinimas.c_str());
         }
      }
      //---------------------------------------------------------------------------
      void __fastcall TForm1::Button4Click(TObject *Sender)
      {
       for(int i = 0; i < this->StringGrid1->RowCount; i++)
        this->StringGrid1->Rows[i]->Clear();

       this->StringGrid1->RowCount = 2;
      }
      //---------------------------------------------------------------------------
      void __fastcall TForm1::Button5Click(TObject *Sender)
      {
       for(int i = 0; i < this->StringGrid2->RowCount; i++)
        this->StringGrid2->Rows[i]->Clear();

       this->StringGrid2->RowCount = 2;
      }
      //---------------------------------------------------------------------------
      void __fastcall TForm1::Button6Click(TObject *Sender)
      {
       if(this->ComboBox1->ItemIndex == -1 || this->ComboBox2->ItemIndex == -1
          || this->ComboBox3->ItemIndex == -1)
       {
        ShowMessage("Pasirinkite visus duomenis!");
        return;
       }
       if(this->ComboBox2->Items->Strings[this->ComboBox2->ItemIndex] > this->ComboBox3->Items-
>Strings[this->ComboBox3->ItemIndex])
       {
        ShowMessage("Blogai pasirinkote datas");
        return;
       }

        this->Label13->Caption = this->ComboBox1->Text;
        AnsiString akcija =  this->ComboBox1->Text;
        this->Series5->Clear();
        this->Series6->Clear();
        this->Series7->Clear();
        this->Series8->Clear();
        this->Series4->Clear();
        this->Series10->Clear();

        this->TabSheet4->TabVisible = true;
        this->PageControl1->Pages[2]->Show();

        this->TabSheet8->TabVisible = false;
        this->TabSheet9->TabVisible = false;

        this->StaticText1->Caption = "";
        this->StaticText2->Caption = "";

        Efekt.spreadV.clear();
        Efekt.sandV.clear();
        Efekt.datosV.clear();

        this->Table1->Active = false;
        this->Table1->DatabaseName = keliasG;
        this->Table1->TableName = lenteleG;
        this->Table1->Active = true;

        this->Table2->Active = false;
        this->Table2->DatabaseName = kelias;
        this->Table2->TableName = "TmpTable.dbf";

         if(this->RadioButton2->Checked)
```

```
          arAkumuliuotas = true;
         else
          arAkumuliuotas = false;

         if(this->CheckBox6->Checked)
          arRGylis = true;
         else
          arRGylis = false;

         if(this->CheckBox7->Checked)
          arKainuPokytis = true;
         else
          arKainuPokytis = false;

         if(this->CheckBox8->Checked)
          arSandoriai = true;
         else
          arSandoriai = false;


        Efekt.UzpildytiSandGridoA(this->StringGrid3);

        for(int i = this->ComboBox2->ItemIndex; i <= this->ComboBox3->ItemIndex; i++)
          Efekt.datosV.push_back(this->ComboBox2->Items->Strings[i]);


        if(this->RadioButton4->Checked)
        {
         this->Timer1->Enabled = true;
         this->Button7->Enabled = false;
         this->Button8->Enabled = false;
         Efekt.Stumti(this->Query1,this->Table1,this->Table2,this->Timer1,this->StaticText1,this-
>StaticText2,
                      this->Button7,this->Button8,this->Series5,this->Series6,this->Series7,this-
>Series8,
                      keliasS,lenteleS,Efekt.datosV,Efekt.spreadV,Efekt.sandV,true,
                      arAkumuliuotas,arRGylis,arKainuPokytis,arSandoriai);
        }
        else
         if(this->RadioButton3->Checked)
         {
          this->Timer1->Enabled = false;
          this->Button7->Enabled = true;
          this->Button8->Enabled = true;
          Efekt.Stumti(this->Query1,this->Table1,this->Table2,this->Timer1,this->StaticText1,this-
>StaticText2,
                       this->Button7,this->Button8,this->Series5,this->Series6,this->Series7,this-
>Series8,
                       keliasS,lenteleS,Efekt.datosV,Efekt.spreadV,Efekt.sandV,
                       true,arAkumuliuotas,arRGylis,arKainuPokytis,arSandoriai);
         }
         else
          if(this->RadioButton5->Checked)
          {
           this->Button7->Enabled = false;
           this->Button8->Enabled = false;
           arRGylis = false;
           Efekt.SkaiciuotiIsKarto(this->Query1,this->Table1,this->Table2,this->StringGrid3,
                                   this->Series5,this->Series6,
                                   this->Series7,this->Series8,this->Series4,this->Series10,
                                   Efekt.datosV,Efekt.spreadV,Efekt.sandV,
                                   keliasS,lenteleS,akcija,
                                   arAkumuliuotas,arRGylis,arKainuPokytis,arSandoriai);
            this->TabSheet8->TabVisible = true;
            this->TabSheet9->TabVisible = true;
          }

      }
      //---------------------------------------------------------------------------
      void __fastcall TForm1::Button8Click(TObject *Sender)
      {
       Efekt.Stumti(this->Query1,this->Table1,this->Table2,this->Timer1,this->StaticText1,this-
>StaticText2,
                    this->Button7,this->Button8,this->Series5,this->Series6,this->Series7,this-
>Series8,
                    keliasS,lenteleS,Efekt.datosV,Efekt.spreadV,Efekt.sandV,true,
                    arAkumuliuotas,arRGylis,arKainuPokytis,arSandoriai);
      }
      //---------------------------------------------------------------------------
```

```
      void __fastcall TForm1::Button7Click(TObject *Sender)
      {
       Efekt.Stumti(this->Query1,this->Table1,this->Table2,this->Timer1,this->StaticText1,
                    this->StaticText2,this->Button7,this->Button8,this->Series5,
                    this->Series6,this->Series7,this->Series8,keliasS,lenteleS,
                    Efekt.datosV,Efekt.spreadV,Efekt.sandV,false,
                    arAkumuliuotas,arRGylis,arKainuPokytis,arSandoriai);
      }
      //---------------------------------------------------------------------------
      void __fastcall TForm1::Timer1Timer(TObject *Sender)
      {
        Efekt.Stumti(this->Query1,this->Table1,this->Table2,this->Timer1,this->StaticText1,this-
>StaticText2,
                    this->Button7,this->Button8,this->Series5,this->Series6,this->Series7,this-
>Series8,
                    keliasS,lenteleS,Efekt.datosV,Efekt.spreadV,Efekt.sandV,true,
                    arAkumuliuotas,arRGylis,arKainuPokytis,arSandoriai);
      }
      //---------------------------------------------------------------------------
      void __fastcall TForm1::Button3Click(TObject *Sender)
      {
       for(int i = 0; i < this->StringGrid3->RowCount; i++)
        this->StringGrid3->Rows[i]->Clear();

       this->StringGrid3->RowCount = 2;
      }
      //---------------------------------------------------------------------------
      void __fastcall TForm1::Uzdaryti1Click(TObject *Sender)
      {
       Close();
      }
      //---------------------------------------------------------------------------

      void __fastcall TForm1::CheckBox9Click(TObject *Sender)
      {
       if(this->CheckBox9->Checked)
        this->CheckBox10->Checked = false;
      }
      //---------------------------------------------------------------------------
      void __fastcall TForm1::CheckBox10Click(TObject *Sender)
      {
       if(this->CheckBox10->Checked)
        this->CheckBox9->Checked = false;
      }
      //---------------------------------------------------------------------------
      void __fastcall TForm1::Button9Click(TObject *Sender)
      {
         AnsiString data,akcija;
       vector <SSkirtumai> sandKainos;
       vector <SSkirtumai> skirtumaiV;

       this->TabSheet1->TabVisible = true;
       this->PageControl1->Pages[4]->Show();
        akcija = this->ComboBox1->Items->Strings[this->ComboBox1->ItemIndex];

         if(this->RadioButton6->Checked)
         {
          for(int i = this->ComboBox2->ItemIndex; i <= this->ComboBox3->ItemIndex; i++)
          {
           data = this->ComboBox2->Items->Strings[i];
           AnsiString uzTekstas = "select Kaina from " + lenteleS + " where Data = '" + data + "'";
           Stac.DuomenuVektorius(this->Query1,sandKainos,keliasS,lenteleS,data,uzTekstas);
          }
          for(int i = 0; i < (int) sandKainos.size() - 1; i++)
           skirtumaiV.push_back(SSkirtumai(sandKainos[i+1].data,sandKainos[i+1].skirtumas -
sandKainos[i].skirtumas));

          AnsiString pavadinimas = akcija + " akcijos įvykusių sandorių kainų kitimo grafikas";
          this->Chart9->Title->Text->SetText(pavadinimas.c_str());
          Stac.VaizduotiDuomenis(this->Series11,sandKainos);

          pavadinimas = akcija + " akcijos įvykusių sandorių kainų skirtumų kitimo grafikas";
          this->Chart10->Title->Text->SetText(pavadinimas.c_str());
          Stac.VaizduotiDuomenis(this->Series12,skirtumaiV);
         }
         else
           if(this->RadioButton7->Checked)
           {
            for(int i = this->ComboBox2->ItemIndex; i <= this->ComboBox3->ItemIndex; i++)
            {
```

```
             data = this->ComboBox2->Items->Strings[i];
             AnsiString uzTekstas = "select Uzd from " + lenteleP + " where Data = '" + data + "' and
Trumpinys = '" + akcija + "'";
             Stac.DuomenuVektorius(this->Query1,sandKainos,keliasP,lenteleP,data,uzTekstas);
            }
            for(int i = 0; i < (int) sandKainos.size() - 1; i++)
              skirtumaiV.push_back(SSkirtumai(sandKainos[i+1].data,sandKainos[i+1].skirtumas -
sandKainos[i].skirtumas));

            AnsiString pavadinimas = akcija + " akcijos uždarymo kainų kitimo grafikas";
            this->Chart9->Title->Text->SetText(pavadinimas.c_str());
            Stac.VaizduotiDuomenis(this->Series11,sandKainos);
            pavadinimas = akcija + " akcijos uždarymo kainų skirtumų kitimo grafikas";
            this->Chart10->Title->Text->SetText(pavadinimas.c_str());
            Stac.VaizduotiDuomenis(this->Series12,skirtumaiV);
           }

      }
      //---------------------------------------------------------------------------
```

Spread.h

```
      //---------------------------------------------------------------------------
      struct SRGylis{
             SRGylis(AnsiString d, AnsiString prL, AnsiString pbL, int piKi, double piKa,  double paKa,
int paKi)
             {data = d; prLaikas = prL; pbLaikas = pbL;
              pirkKaina = piKa; pirkKiekis = piKi;
              pardKaina = paKa; pardKiekis = paKi;}

             AnsiString data, prLaikas,pbLaikas;
             int pirkKiekis,pardKiekis;
             double pirkKaina,pardKaina;

        };
      struct SSand{
            SSand(AnsiString l, double k)
            {laikas = l; kaina = k;}

            AnsiString laikas;
            double kaina;
      };


      class TSpread
      {
        public:
          void PaprastasSpredas(vector<SRGylis>&,vector<double>&,vector<double>&);  //skaičiuoja
"quoted spread"
          void IrasytiGyli(TQuery*,vector<SRGylis>&,AnsiString,AnsiString); //surašo gylio duomenis į
vektorių
          AnsiString PridetiSekundes(AnsiString,int); //prie nurodyto laiko prideda nurodytą sekundžių
kiekį
          void Vidurkis(vector<double>&,double&); //skaičiuoja pateiktų duomenų vidurkį
          void EfektyvusSpredas(vector<SRGylis>&,vector<SSand>&,vector<double>&,vector<double>&);
//skaičiuoja  "effective spread"
          void UzfiksuotasSpredas(vector<SSand>&,vector<double>&); //skaičiuoja sandorių kainų skirtumą
          void IrasytiSandorius(TQuery*,vector<SSand>&,AnsiString,AnsiString,bool&); //sudaro sandorių
vektorių
          void RollSpredas(vector<double>&,double&,bool&); //skaičiuoja "Roll spread"
          void Antraste(TStringGrid*); //užpildo rezultatų lentelės antraštę
          void IsvestiRezultatus(TStringGrid*,TCheckBox*,TCheckBox*,TCheckBox*,TCheckBox*,
                                 vector<double>&,vector<double>&,vector<double>&,double,bool,
                                 AnsiString,AnsiString,AnsiString); //išveda rezultatus
          void VaizduotiIvercius(TLineSeries*,vector<SSand>&);  //grafike vaizduoja Roll įverčius
      };
      //---------------------------------------------------------------------------
```

Spread.cpp

```
      //---------------------------------------------------------------------------
      void TSpread::IrasytiGyli(TQuery*uzklausa,vector<SRGylis>&gylis,
                                AnsiString kelias,AnsiString uzTekstas)
      {

       AnsiString prLaikas = "";
       AnsiString pbLaikas = "";
```

```
       try
       {
        uzklausa->DatabaseName = kelias;
        uzklausa->Close();
        uzklausa->SQL->Clear();
        uzklausa->SQL->Add(uzTekstas);
        uzklausa->Open();
        uzklausa->First();


        while(!uzklausa->Eof)
        {
         if(prLaikas != uzklausa->FieldByName(uzklausa->Fields->Fields[1]->FieldName)->AsString
            && pbLaikas != uzklausa->FieldByName(uzklausa->Fields->Fields[2]->FieldName)->AsString)
         {

           gylis.push_back(SRGylis(uzklausa->FieldByName(uzklausa->Fields->Fields[0]->FieldName)-
>AsString,
                           uzklausa->FieldByName(uzklausa->Fields->Fields[1]->FieldName)->AsString,
                           uzklausa->FieldByName(uzklausa->Fields->Fields[2]->FieldName)->AsString,
                           uzklausa->FieldByName(uzklausa->Fields->Fields[3]->FieldName)->AsInteger,
                           uzklausa->FieldByName(uzklausa->Fields->Fields[4]->FieldName)->AsFloat,
                           uzklausa->FieldByName(uzklausa->Fields->Fields[5]->FieldName)->AsFloat,
                           uzklausa->FieldByName(uzklausa->Fields->Fields[6]->FieldName)->AsInteger));

           prLaikas = uzklausa->FieldByName(uzklausa->Fields->Fields[1]->FieldName)->AsString;
           pbLaikas = uzklausa->FieldByName(uzklausa->Fields->Fields[2]->FieldName)->AsString;
         }

          uzklausa->Next();
        }
        uzklausa->Close();

       }
       catch (...)
       {
         ShowMessage("Duomenu nuskaityti nepavyko.");
       }

      }
      //---------------------------------------------------------------------------
      void TSpread::IrasytiSandorius(TQuery*uzklausa,vector<SSand>&sandoriai,
                                     AnsiString kelias,AnsiString uzTekstas,bool&arViskasGerai)
      {

       try
       {
        uzklausa->DatabaseName = kelias;
        uzklausa->Close();
        uzklausa->SQL->Clear();
        uzklausa->SQL->Add(uzTekstas);
        uzklausa->Open();
        uzklausa->First();


        while(!uzklausa->Eof)
        {
          sandoriai.push_back(SSand(uzklausa->FieldByName(uzklausa->Fields->Fields[0]->FieldName)-
>AsString,
                                    uzklausa->FieldByName(uzklausa->Fields->Fields[1]->FieldName)-
>AsFloat));
          uzklausa->Next();
        }
        uzklausa->Close();

        arViskasGerai = true;
       }
       catch (...)
       {
         ShowMessage("Įvyko klaida. Kai kurių sandorių duomenų nuskaityti nepavyko. Dėl šios
priežasties ne visi pasirinkti rodikliai bus apskaičiuoti.");
         arViskasGerai = false;
        }

      }
      //---------------------------------------------------------------------------
      void
TSpread::PaprastasSpredas(vector<SRGylis>&duom,vector<double>&qSpread,vector<double>&qSpreadV)
```

```
{
 AnsiString prLaikas = "10:00:00";
 AnsiString pbLaikas = "13:49:59";
 int intervalas = 10;  //kas kiek sekundžių imam spredus
 AnsiString laikas = prLaikas;

 while(prLaikas <= laikas && laikas <= pbLaikas)
 {
  for(int i = 0; i < (int) duom.size(); i++)
  {
    if(duom[i].prLaikas <= laikas && laikas <= duom[i].pbLaikas)
    {
     qSpread.push_back(duom[i].pardKaina - duom[i].pirkKaina);
     break;
    }
    else
     if(i == (int) duom.size() - 1)
     {
      for(int j = 0; j < (int) duom.size(); j++)
       if(laikas < duom[j].prLaikas && laikas < duom[j].pbLaikas)
       {
        if(j != 0)
         qSpread.push_back(duom[j-1].pardKaina - duom[j-1].pirkKaina);
        else
          qSpread.push_back(duom[j].pardKaina - duom[j].pirkKaina);
        break;
       }
     }
  }
  laikas = PridetiSekundes(laikas,intervalas);
 }
}
//---------------------------------------------------------------------------
AnsiString TSpread::PridetiSekundes(AnsiString laikas,int sek)
{
  int pglSek;
  div_t x,y;
  AnsiString val,min,sekk;

  pglSek = StrToInt(laikas.SubString(1,2))*3600
         + StrToInt(laikas.SubString(4,2))*60
         + StrToInt(laikas.SubString(7,2)) + sek;

  x = div(pglSek,3600);
  y = div(x.rem,60);

  if(x.quot < 10)
   val = "0" + IntToStr(x.quot);
  else
   val = IntToStr(x.quot);

  if(y.quot < 10)
   min = "0" + IntToStr(y.quot);
  else
   min = IntToStr(y.quot);

  if(y.rem < 10)
   sekk = "0" + IntToStr(y.rem);
  else
   sekk = IntToStr(y.rem);


  AnsiString nLaikas = val + ":" + min + ":" + sekk;

  return nLaikas;
}
//---------------------------------------------------------------------------
void TSpread::Vidurkis(vector<double>&duom,double &vidurkis)
{
 double suma = 0;

 for(int i = 0; i < (int) duom.size(); i++)
  suma = suma + duom[i];
 if(duom.size() != 0)
   vidurkis = (double) suma / duom.size();
}
//---------------------------------------------------------------------------
void TSpread::EfektyvusSpredas(vector<SRGylis>&duom,vector<SSand>&sandoriai,
                               vector<double>&eSpread,
                               vector<double>&eSpreadV)
```

```
{

 double pglEspread;

 for(int i = 0; i < (int) sandoriai.size(); i++)
  for(int j = 0; j < (int) duom.size(); j++)
   if(duom[j].prLaikas <= sandoriai[i].laikas && sandoriai[i].laikas <= duom[j].pbLaikas)
   {
      pglEspread = 2*fabs(sandoriai[i].kaina - (duom[j].pardKaina + duom[j].pirkKaina)/2);
      eSpread.push_back(pglEspread);
      break;
   }

}
//---------------------------------------------------------------------------
void TSpread::UzfiksuotasSpredas(vector<SSand>&sandoriai,vector<double>&uSpreadasV)
{
  for(int i = 0; i < (int) sandoriai.size() - 1; i++)
   uSpreadasV.push_back(sandoriai[i + 1].kaina - sandoriai[i].kaina);
}
//---------------------------------------------------------------------------
void TSpread::RollSpredas(vector<double>&sandSkirtumai,double& ivertis,bool& arNeigiamas)
{
  double cov,sandVid,vidurkis1,vidurkis2;
  double suma1 = 0;
  double suma2 = 0;
  double suma = 0;

  if(sandSkirtumai.size() != 0 && sandSkirtumai.size() != 1)
  {
   for(int i = 0; i < (int) sandSkirtumai.size() -1; i++)
   {
    suma1 = suma1 + sandSkirtumai[i];
    suma2 = suma2 + sandSkirtumai[i+1];
   }

   vidurkis1 = suma1 / (sandSkirtumai.size() - 1);
   vidurkis2 = suma2 / (sandSkirtumai.size() - 1);

   for(int j = 1; j < (int) sandSkirtumai.size(); j++)
    suma = suma + sandSkirtumai[j]*sandSkirtumai[j-1];

   cov = (suma / (sandSkirtumai.size() - 1)) - vidurkis1*vidurkis2;


   if(cov <= 0)
   {
    ivertis = sqrt(-4*cov);
    arNeigiamas = false;
   }
   else
    arNeigiamas = true;
  }
  else
    arNeigiamas = true;
}
```

Stacionarumas.h

```
//---------------------------------------------------------------------------
struct SKvantiliai{
       SKvantiliai(double a, double b, double c)
            {alfa = a; alfa1_2 = b; alfa2 = c;}
  double alfa;
  double alfa1_2;
  double alfa2;
};
struct SIntervalas{
       SIntervalas(double af, int a, int b)
       {alfa = af; i1 = a; i2 = b;}
      double alfa;
      int i1,i2;
      };

struct SLaikasKaina{
    SLaikasKaina(AnsiString l, double ll)
    {laikas = l; duom = ll;}
    AnsiString laikas;
    double duom;
```

```
};

struct SSkirtumai{
    SSkirtumai(AnsiString d, double s)
    {data = d; skirtumas = s;}
    AnsiString data;
    double skirtumas;
};

class TStacionarumas{

  public:
    void
DuomenuVektorius(TQuery*,vector<SSkirtumai>&,AnsiString,AnsiString,AnsiString,AnsiString); //sudaro
duomenų vektorių
     // void
DuomenuParuosimas(TLineSeries*,TMemo*,TCheckBox*,vector<SLaikasKaina>&,vector<double>&,vector<double>&)
; //paruošia duomenis RAT testui
      void VaizduotiDuomenis(TLineSeries*,vector<SSkirtumai>&); //vaizduoja duomenis grafiku
      void VaizduotiVidurkius(TLineSeries*,vector<double>&); //vaizduoja duomenis grafiku
      void
SkaiciuotiVidurkius(vector<SSkirtumai>&,vector<double>&,vector<double>&,vector<SSkirtumai>&);
//apskaičiuoja EX ir E(X^2)
      void RAT(TStringGrid*,vector<double>&,AnsiString,AnsiString,AnsiString,AnsiString); //Reverse
arrangement test
      void
IsvestiRezultatus(TStringGrid*,vector<SIntervalas>&,AnsiString,AnsiString,AnsiString,AnsiString,int);//
išveda rezultatus į StringGrid
      void GridoAntraste(TStringGrid*); //užpildo StringGrid antraštę
};
//---------------------------------------------------------------------------
```

Stacionarumas.cpp

```
//---------------------------------------------------------------------------

void TStacionarumas::DuomenuVektorius(TQuery*uzklausa,vector <SSkirtumai> &duom,
                                      AnsiString keliasS,AnsiString lenteleS,AnsiString data,
                                      AnsiString uzTekstas)
{
  double kaina;

 try
 {
  uzklausa->Close();
  uzklausa->DatabaseName = keliasS;
  uzklausa->SQL->Clear();
  uzklausa->SQL->Add(uzTekstas);
  uzklausa->Open();
  uzklausa->First();

  while(!uzklausa->Eof)
  {
    kaina = uzklausa->FieldByName(uzklausa->Fields->Fields[0]->FieldName)->AsFloat;
    if(kaina != 0)
     duom.push_back(SSkirtumai(data,kaina));
    uzklausa->Next();
  }

  uzklausa->Close();

 }
 catch (...)
 {
   ShowMessage("Duomenu nuskaityti nepavyko.");
 }
}
//---------------------------------------------------------------------------
void TStacionarumas::VaizduotiDuomenis(TLineSeries*series,vector<SSkirtumai>&duom)
{
  series->Clear();
  for(int i = 0; i < (int) duom.size(); i++)
   series->AddXY(i,duom[i].skirtumas,IntToStr(i+1),clTeeColor);

}
//---------------------------------------------------------------------------
void TStacionarumas::VaizduotiVidurkius(TLineSeries*series,vector<double>&duom)
{
```

```
  series->Clear();
  for(int i = 0; i < (int) duom.size(); i++)
   series->AddXY(i,duom[i],IntToStr(i+1),clTeeColor);

}
//---------------------------------------------------------------------------
void TStacionarumas::SkaiciuotiVidurkius(vector<SSkirtumai>&duom,vector<double>&meanX,
                                         vector<double>&meanX2,vector<SSkirtumai>&skirtumaiV)
{
  double suma;
  double suma2;
  int elKiek;


  if(duom.size() == 0)
   return;

  for(int i = 0; i < (int) duom.size() - 1; i++)
   skirtumaiV.push_back(SSkirtumai(duom[i+1].data,duom[i+1].skirtumas - duom[i].skirtumas));

   for(int i = 0;  i < (int) skirtumaiV.size(); i++)
   {
    if(i == 0)
    {
     suma = 0;
     suma2 = 0;
     elKiek = 0;
     suma = suma + skirtumaiV[i].skirtumas;
     suma2 = suma2 + skirtumaiV[i].skirtumas*skirtumaiV[i].skirtumas;
     elKiek++;
     if(i == (int) skirtumaiV.size() - 1)
      {
       meanX.push_back(suma / elKiek);
       meanX2.push_back(suma2 / elKiek);
      }
    }
    else
     if(skirtumaiV[i].data == skirtumaiV[i - 1].data)
     {
      suma = suma + skirtumaiV[i].skirtumas;
      suma2 = suma2 + skirtumaiV[i].skirtumas*skirtumaiV[i].skirtumas;
      elKiek++;
      if(i == (int) skirtumaiV.size() - 1)
      {
       meanX.push_back(suma / elKiek);
       meanX2.push_back(suma2 / elKiek);
      }
     }
     else
     {
      meanX.push_back(suma / elKiek);
      meanX2.push_back(suma2 / elKiek);
      suma = 0;
      suma2 = 0;
      elKiek = 0;
      suma = suma + skirtumaiV[i].skirtumas;
      suma2 = suma2 + skirtumaiV[i].skirtumas*skirtumaiV[i].skirtumas;
      elKiek++;
      if(i == (int) skirtumaiV.size() - 1)
      {
       meanX.push_back(suma / elKiek);
       meanX2.push_back(suma2 / elKiek);
      }
     }
   }
}
//---------------------------------------------------------------------------
void TStacionarumas::RAT(TStringGrid*gridas,vector <double> &duom,AnsiString akcija,
                         AnsiString data1,AnsiString data2,AnsiString dydis)
{
 int N = duom.size();
 int A = 0;

 for(int i = 0; i < N - 1; i++)
  for(int j = i + 1; j < N; j++)
   if(duom[i] > duom[j])
     A++;

  //--- hipotezės priėmimo srities formavimas-------
  double EX = N*(N - 1)/4;
```

```
        double DX = N*(2*N + 5)*(N - 1)/72;
        vector <SKvantiliai> z;
        z.push_back(SKvantiliai(0.01,-2.575829303548910,2.575829303548920));
        z.push_back(SKvantiliai(0.05,-1.959963984540050,1.959963984540050));
        vector <SIntervalas> intervalas;
        for(int i = 0; i < (int) z.size(); i++)
        {
         int i1,i2;
         i1 = int(z[i].alfa1_2 * sqrt(DX) + EX);
         i2 = int(z[i].alfa2 * sqrt(DX) + EX);
         intervalas.push_back(SIntervalas(z[i].alfa,i1,i2));
        }
       //---------------------------------------------------------

        IsvestiRezultatus(gridas,intervalas,akcija,data1,data2,dydis,A);
      }
```

Efektyvumas.h

```
      //---------------------------------------------------------------------------
      struct SKainaKiekis {
              SKainaKiekis(double ka, double ki)
              {kaina = ka; kiekis = ki;}
        double kaina,kiekis;
      };

      struct SKainaKiekisC{
              SKainaKiekisC(double ka, double ki, TColor cl)
              {kaina = ka; kiekis = ki; color = cl;}
        double kaina,kiekis;
        TColor color;
      };

      struct SSpread{
         SSpread(AnsiString l, double s)
         {laikas = l; spread = s;}
        AnsiString laikas;
        double spread;
      };

      struct SSandoriai{
       SSandoriai(AnsiString l, double ka, int ki, int in)
       {laikas = l; kaina = ka; kiekis = ki; inic = in;}
       AnsiString laikas;
       double kaina;
       int kiekis,inic;
      };

      class TEfektyvumas
      {
       public:

         vector<AnsiString> datosV;
         vector <SSpread> spreadV; //kaupia vidutinius spreadus vaizdavimui
         vector <SSandoriai> sandV; //kaupia visus sandorius vaizdavimui

         void FiltruotiLentele(TTable*,AnsiString); //filtruoja lentelę pagal nurodytus kriterijus
         void Stumti(TQuery*,TTable*,TTable*,TTimer*,TStaticText*,TStaticText*,TButton*,TButton*,
                     TAreaSeries*,TLineSeries*,TPointSeries*,TPointSeries*,
                     AnsiString,AnsiString,
                     vector<AnsiString>&,vector<SSpread>&,
                     vector<SSandoriai>&, bool,bool,bool,bool,bool); //paslenka laiką į priekį arba
atgal
         void ApskaiciuotiDataLaika(TTimer*,TStaticText*,TStaticText*,TButton*,TButton*,
                                    vector<AnsiString>&,bool,AnsiString,AnsiString,
                                    AnsiString,AnsiString); // apskaičiuoja laiką

         void PerkeltiLentele(TTable*,TTable*); //perkelia lentelę į kitą
         void IrasytiKainasKiekius(TQuery*,vector<SKainaKiekis>&,AnsiString,AnsiString,AnsiString); //į
vektorių surašo rinkos gylio lentelės kainas
         void VaizduotiDuomenis1(TAreaSeries*,vector<SKainaKiekisC>&,bool); //vaizduoja duomenis
grafiku  (rinkos gylis)
         void
ApskaiciuotiSpreada(TQuery*,AnsiString,AnsiString,AnsiString,AnsiString,vector<SSpread>&,AnsiString);
//apskaičiuoja vidutinį spreadą
         void VaizduotiDuomenis2(TLineSeries*,vector<SSpread>&); //vaizduoja duomenis grafiku (spredą)
```

```
        void SandoriuSarasas(TQuery*,AnsiString,vector<SSandoriai>&,AnsiString);// sudaro įvykusių
sandorių sąrašą
        void NustatytiIniciatoriu(TTable*,vector<SSandoriai>&,AnsiString); //nustato įvykusio sandorio
iniciatorių
        void PiestiGrafikaSI(TPointSeries*series,vector <SSandoriai> sandoriai,bool kainos);//piešia
sandorių grafiką (kainas/kiekius + iniciatorius)
        void SkaiciuotiIsKarto(TQuery*,TTable*,TTable*,TStringGrid*,TAreaSeries*,TLineSeries*,

TPointSeries*,TPointSeries*,TBarSeries*,TBarSeries*,vector<AnsiString>&,vector<SSpread>&,
                               vector<SSandoriai>&,AnsiString,AnsiString,AnsiString,bool, bool,
                               bool,bool); //skaičiuoja neišvedant tarpinių rezultatų
        void TarpiniaiSkaiciavimai(TQuery*,TTable*,TAreaSeries*,TLineSeries*,TPointSeries*,

TPointSeries*,vector<SSpread>&,vector<SSandoriai>&,AnsiString,AnsiString,AnsiString,AnsiString,bool,boo
l,bool,bool,bool);
        void UzpildytiSandGridoA(TStringGrid*);    //užpildo sandorių lentelės antraštę

        void SandoriuInfo(TStringGrid*,AnsiString,AnsiString,AnsiString,int,double,
double,int,double); //išveda sandorių info
        void InicProc(vector<SSandoriai>&,double &,double&); //suskaičiuoja, kiek sandorių iniciavo
viena, kiek kita pusė


        TSpread Spread; //klasės "TSpread" objektas

    };
    //---------------------------------------------------------------------------
```

Efektyvumas.cpp

```
//---------------------------------------------------------------------------
void TEfektyvumas::FiltruotiLentele(TTable*table,AnsiString filtras)
{
 table->Active = false;
 table->Filtered = false;
 table->Filter = filtras;
 table->Filtered = true;
 table->Active = true;
}
//---------------------------------------------------------------------------
void TEfektyvumas::Stumti(TQuery*query,TTable*table1,TTable*table2,
                          TTimer*timer,
                          TStaticText*st1,TStaticText*st2,
                           TButton*atgalB,TButton*pirmynB,
                           TAreaSeries*seriesPiPa,TLineSeries*seriesSp,
                           TPointSeries*seriesSKa,TPointSeries*seriesSKi,
                           AnsiString keliasS,AnsiString lenteleS,
                           vector<AnsiString>&datos,vector<SSpread>& spread,
                           vector<SSandoriai>&sandoriaiV,
                           bool pirmyn,bool arAkumuliuotas,bool arRGylis,
                           bool arKainuPokytis,bool arSandoriai)
{


 AnsiString data = st1->Caption;
 AnsiString laikas = st2->Caption;

 AnsiString filtras;
 AnsiString dataP = table1->Fields->Fields[0]->FieldName;

 AnsiString pradLaikas = "10:00";
 AnsiString pabLaikas = "13:49";
 AnsiString pradData = datos.front();
 AnsiString pabData = datos.back();

 int inicPa,inicPi;

 bool arStumti = true;


 if(st1->Caption == "" && st2->Caption == "")
 {
  st1->Caption = pradData;
  st2->Caption = pradLaikas;
  atgalB->Enabled = false;
  pirmynB->Enabled = true;
  arStumti = false;
 }
```

```
       if(arStumti)

ApskaiciuotiDataLaika(timer,st1,st2,atgalB,pirmynB,datos,pirmyn,pradLaikas,pabLaikas,pradData,pabData);


       if(data != st1->Caption)
       {
        data = st1->Caption;
        filtras =  dataP + " = '" +  data + "'";
        FiltruotiLentele(table1,filtras);
        table2->Active = false;
        table2->EmptyTable();
        PerkeltiLentele(table1,table2);
        table2->Active = true;
       }

       laikas = st2->Caption;
       TarpiniaiSkaiciavimai(query,table2,seriesPiPa,seriesSp,seriesSKa,seriesSKi,
                             spread,sandoriaiV,keliasS,lenteleS,data,
                             laikas,arAkumuliuotas,arRGylis,arKainuPokytis,arSandoriai,true);

      }
      //---------------------------------------------------------------------------
      void TEfektyvumas::ApskaiciuotiDataLaika(TTimer*timer,TStaticText*st1,TStaticText*st2,
                                               TButton*atgalB,TButton*pirmynB,
                                               vector<AnsiString>&datos,bool pirmyn,
                                               AnsiString pradLaikas,AnsiString pabLaikas,
                                               AnsiString pradData, AnsiString pabData)
      {

        int datosIndex;

        AnsiString data = st1->Caption;
        AnsiString laikas = st2->Caption;


        for(int i = 0; i < (int) datos.size(); i++)
         if(datos[i] == data)
          datosIndex = i;

        if(pirmyn)
        {
         laikas = Spread.PridetiSekundes(laikas + ":00",60);
         if(laikas.SubString(1,5) <= pabLaikas)
         {
          st2->Caption = laikas.SubString(1,5);
          atgalB->Enabled = true;
          if(laikas.SubString(1,5) == pabLaikas && data == pabData)
          {
           pirmynB->Enabled = false;
           timer->Enabled = false;
          }
         }
         else
          if(data != pabData)
          {
           st1->Caption = datos[datosIndex + 1];
           st2->Caption = pradLaikas;
          }
          else
          {
           pirmynB->Enabled = false;
           timer->Enabled = false;
          }
        }
        else
        {
         laikas = Spread.PridetiSekundes(laikas + ":00",-60);
         if(laikas.SubString(1,5) >= pradLaikas)
         {
          st2->Caption = laikas.SubString(1,5);
          pirmynB->Enabled = true;
          if(laikas.SubString(1,5) == pradLaikas && data == pradData)
            atgalB->Enabled = false;
         }
         else
          if(data != pradData)
          {
           st1->Caption = datos[datosIndex - 1];
```

```
          st2->Caption = pabLaikas;
         }
         else
          atgalB->Enabled = false;
       }
      }
      //---------------------------------------------------------------------------
      void TEfektyvumas::VaizduotiDuomenis1(TAreaSeries*seriesPiPa,vector<SKainaKiekisC>&duom,
                                            bool akumuliuotas)
      {
        seriesPiPa->Clear();
        double kiek = 0;
        int index;
        if(akumuliuotas)
         for(int i = 0; i < (int) duom.size(); i++)
         {
          kiek = 0;
          if(i == 0)
           index = 0;
          else
           if(duom[i - 1].color != duom[i].color)
           {
            index = i;
            seriesPiPa->AddNullXY(duom[i].kaina,duom[i].kiekis,FloatToStr(duom[i].kaina));
           }

          for(int ii = index; ii <= i; ii++)
           kiek = kiek + duom[ii].kiekis;

          seriesPiPa->AddXY(duom[i].kaina,kiek,FloatToStr(duom[i].kaina),duom[i].color);
         }
        else
         for(int i = 0; i < (int) duom.size(); i++)
          if(i != 0 && duom[i - 1].color != duom[i].color)
          {
           seriesPiPa->AddNullXY(duom[i].kaina,duom[i].kiekis,FloatToStr(duom[i].kaina));
           seriesPiPa->AddXY(duom[i].kaina,duom[i].kiekis,FloatToStr(duom[i].kaina),duom[i].color);
          }
          else
           seriesPiPa->AddXY(duom[i].kaina,duom[i].kiekis,FloatToStr(duom[i].kaina),duom[i].color);

      }
      //---------------------------------------------------------------------------
      void TEfektyvumas::ApskaiciuotiSpreada(TQuery*uzklausa,AnsiString kelias,AnsiString
lentele,AnsiString laikas,
                                             AnsiString uzTekstas,vector<SSpread>&spreadas,AnsiString
data)
      {
       vector<AnsiString> laikai;
       AnsiString uzTekstas2;
       try
       {
        uzklausa->Close();
        uzklausa->DatabaseName = kelias;
        uzklausa->SQL->Clear();
        uzklausa->SQL->Add(uzTekstas);
        uzklausa->Open();
        uzklausa->First();
        while(!uzklausa->Eof)
        {
         laikai.push_back(uzklausa->FieldByName(uzklausa->Fields->Fields[0]->FieldName)->AsString);
         uzklausa->Next();
        }
        uzklausa->Close();


         double suma = 0;
         for(int i = 0; i < (int) laikai.size(); i++)
         {
          uzTekstas2 = "select Pard_kaina,Pirk_kaina from " + lentele
                       + " where data = '" + data + "' and Pr_laikas = '" + laikai[i] + "'";

          uzklausa->Close();
          uzklausa->DatabaseName = kelias;
          uzklausa->SQL->Clear();
          uzklausa->SQL->Add(uzTekstas2);
          uzklausa->Open();
          uzklausa->First();

          if(uzklausa->FieldByName(uzklausa->Fields->Fields[0]->FieldName)->AsFloat != 0
```

```
            && uzklausa->FieldByName(uzklausa->Fields->Fields[1]->FieldName)->AsFloat != 0)
          suma = suma + uzklausa->FieldByName(uzklausa->Fields->Fields[0]->FieldName)->AsFloat
                      - uzklausa->FieldByName(uzklausa->Fields->Fields[1]->FieldName)->AsFloat;

        }

       uzklausa->Close();
       if(laikai.size() != 0)
       {
         double pglVid = suma/laikai.size();
         int pglVidInt;
         pglVid = pglVid * 100;
         pglVidInt = floor(pglVid + 0.5);
         pglVid = (double) pglVidInt/100;
         spreadas.push_back(SSpread(laikas,pglVid));
       }
       else
         spreadas.push_back(SSpread(laikas,-0.01));  //langas duomenyse
      }
      catch (...)
      {
        ShowMessage("Duomenu nuskaityti nepavyko.");
      }
     }
     //---------------------------------------------------------------------------
     void TEfektyvumas::VaizduotiDuomenis2(TLineSeries*series,vector<SSpread>&duom)
     {
       series->Clear();

        for(int i = 0; i < (int) duom.size(); i++)
         series->AddXY(i,duom[i].spread,duom[i].laikas,clTeeColor);
     }
     //---------------------------------------------------------------------------
     void TEfektyvumas::NustatytiIniciatoriu(TTable*table2,vector<SSandoriai>&sandoriai,AnsiString
data)
     {
      AnsiString filtras;
       // +1 - sandori inicijavo pirkejas
      // -1 -sandori inicijavo pardavejas

      table2->Active = true;
      AnsiString pirkKainaP = table2->Fields->Fields[5]->FieldName;
      AnsiString pardKainaP = table2->Fields->Fields[7]->FieldName;

       for(int i = 0; i < (int) sandoriai.size(); i++)
       {
         filtras =  "data = '" + data + "' and Pr_laikas <= '" +  sandoriai[i].laikas + "' and
Pb_laikas >= '" + sandoriai[i].laikas + "'";
         FiltruotiLentele(table2,filtras);
         table2->First();
         while(!table2->Eof)
         {
          if(table2->FieldByName(pirkKainaP)->AsFloat == sandoriai[i].kaina)
          {
           sandoriai[i].inic = -1; //inicijavo pardavėjas
           break;
          }
          else
           if (table2->FieldByName(pardKainaP)->AsFloat == sandoriai[i].kaina)
           {
            sandoriai[i].inic = 1; //inicijavo pirkėjas
            break;
           }
         table2->Next();
         }
       }
     }
     //---------------------------------------------------------------------------
     void TEfektyvumas::PiestiGrafikaSI(TPointSeries*series,vector <SSandoriai> sandoriai,bool kainos)
     {
        series->Clear();
        for(int i = 0; i < (int) sandoriai.size(); i++)
        {
         if(kainos)
          series->AddXY(i,sandoriai[i].kaina,sandoriai[i].laikas,clTeeColor);
         else
          series->AddXY(i,sandoriai[i].kiekis,sandoriai[i].laikas,clTeeColor);

         if(sandoriai[i].inic == -1)
           series->ColorRange(series->XValues,i,i,clGreen);
```

```
    else
     if(sandoriai[i].inic == 1)
      series->ColorRange(series->XValues,i,i,clRed);
     else
      series->ColorRange(series->XValues,i,i,clBlack);
   }
}
//---------------------------------------------------------------------------
void TEfektyvumas::SkaiciuotiIsKarto(TQuery*query,TTable*table1,TTable*table2,
                                     TStringGrid*gridasSand,
                                     TAreaSeries*seriesPiPa,
                                     TLineSeries*seriesSp,TPointSeries*seriesSKa,
                                     TPointSeries*seriesSKi,TBarSeries*seriesPi,
                                     TBarSeries*seriesPa,
                                     vector<AnsiString>&datos,vector<SSpread>&spread,
                                     vector<SSandoriai>&sandoriaiV,
                                     AnsiString keliasS,AnsiString lenteleS,
                                     AnsiString akcija,
                                     bool arAkumuliuotas, bool arRGylis,
                                     bool arKainuPokytis, bool arSandoriai)
{

 vector<SSandoriai> sandoriai;

 AnsiString data;
 AnsiString laikas;

 double inicPad, inicPid;
 int dienuSk = 0;

 AnsiString filtras,uzTekstas,uzTekstas2;
 AnsiString dataP = table1->Fields->Fields[0]->FieldName;

 AnsiString pradLaikas = "10:00";
 AnsiString pabLaikas = "13:49";
 AnsiString pradData = datos.front();
 AnsiString pabData = datos.back();


 for(int i = 0; i < (int) datos.size(); i++)
 {
  dienuSk++;
  filtras =  dataP + " = '" +  datos[i] + "'";
  FiltruotiLentele(table1,filtras);
  table2->Active = false;
  table2->EmptyTable();
  PerkeltiLentele(table1,table2);
  table2->Active = true;
  laikas = pradLaikas;
  sandoriai.clear();
  while(laikas <= pabLaikas)
  {
   TarpiniaiSkaiciavimai(query,table2,seriesPiPa,seriesSp,seriesSKa,seriesSKi,
                         spread,sandoriai,keliasS,lenteleS,datos[i],
                         laikas,arAkumuliuotas,arRGylis,arKainuPokytis,arSandoriai,false);

   laikas = Spread.PridetiSekundes(laikas + ":00",60).SubString(1,5);
  }
  for(int i = 0; i < (int) sandoriai.size(); i++)
   sandoriaiV.push_back(sandoriai[i]);

  if(arKainuPokytis)
   VaizduotiDuomenis2(seriesSp,spread);
  if(arSandoriai)
  {
   data = datos[i];
   InicProc(sandoriai,inicPid,inicPad);
   seriesPa->AddBar(inicPad,data,clGreen);
   seriesPi->AddBar(inicPid,data,clRed);
  }
 }
 if(arSandoriai)
 {
  PiestiGrafikaSI(seriesSKa,sandoriaiV,true);
  PiestiGrafikaSI(seriesSKi,sandoriaiV,false);
  InicProc(sandoriaiV,inicPid,inicPad);

  double apyvarta = 0;
  for(int ii = 0; ii < (int) sandoriaiV.size(); ii++)
   apyvarta = apyvarta + sandoriaiV[ii].kaina * sandoriaiV[ii].kiekis;
```

```
SandoriuInfo(gridasSand,akcija,pradData,pabData,dienuSk,inicPid,inicPad,sandoriaiV.size(),apyvarta);
       }
      }
      //---------------------------------------------------------------------------
      void TEfektyvumas::InicProc(vector<SSandoriai>&sandoriai,double & inicPid,double&inicPad)
      {
        int inicPi = 0;
        int inicPa = 0;

        for(int i = 0; i < (int) sandoriai.size(); i++)
        {
         if(sandoriai[i].inic == 1)
          inicPi++;
         else if (sandoriai[i].inic == -1)
          inicPa++;
        }

        inicPid = 0;
        inicPad = 0;
        if(sandoriai.size() != 0)
        {
         inicPid =  (double)inicPi*100/sandoriai.size();
         inicPad =  (double)inicPa * 100/sandoriai.size();
        }
      }
      //---------------------------------------------------------------------------
      void TEfektyvumas::TarpiniaiSkaiciavimai(TQuery*query,TTable*table2,
                                               TAreaSeries*seriesPiPa,
                                               TLineSeries*seriesSp,TPointSeries*seriesSKa,
                                               TPointSeries*seriesSKi,vector<SSpread>&spread,
                                               vector<SSandoriai>&sandoriaiV,
                                               AnsiString keliasS,AnsiString lenteleS,
                                               AnsiString data,AnsiString laikas,
                                               bool arAkumuliuotas, bool arRGylis,
                                               bool arKainuPokytis, bool arSandoriai,
                                               bool arVaizduoti)
      {

       AnsiString uzTekstas,uzTekstas2;
       vector<SKainaKiekis> pirkimai;
       vector<SKainaKiekis> pardavimai;
       vector<SSandoriai> sandoriai;
       vector<SKainaKiekisC> pirkPard;

       if(arRGylis)
       {
        uzTekstas = "select Pirk_kaina,Pirk_kiek from " + table2->TableName
                    + " where data = '" + data + "' and Pr_laikas <= '" + laikas + ":59' and Pb_laikas
>= '"
                    + laikas + ":00' and Pirk_kaina <> 0 order by Pirk_kaina desc";

        uzTekstas2 = "select distinct Pr_laikas from " + table2->TableName
                     + " where data = '" + data + "' and Pr_laikas <= '" + laikas + ":59' and Pb_laikas
>= '"
                     + laikas + ":00'";
        IrasytiKainasKiekius(query,pirkimai,table2->DatabaseName,uzTekstas,uzTekstas2);
        for(int i = 0; i < (int) pirkimai.size(); i++)
         pirkPard.push_back(SKainaKiekisC(pirkimai[i].kaina,pirkimai[i].kiekis,clGreen));

        uzTekstas = "select Pard_kaina,Pard_kiek from " + table2->TableName
                    + " where data = '" + data + "' and Pr_laikas <= '" + laikas + ":59' and Pb_laikas
>= '"
                    + laikas + ":00' and Pard_kaina <> 0 order by Pard_kaina";

        IrasytiKainasKiekius(query,pardavimai,table2->DatabaseName,uzTekstas,uzTekstas2);
        for(int i = 0; i < (int) pardavimai.size(); i++)
         pirkPard.push_back(SKainaKiekisC(pardavimai[i].kaina,pardavimai[i].kiekis,clRed));

        VaizduotiDuomenis1(seriesPiPa,pirkPard,arAkumuliuotas);
       }
       if(arKainuPokytis)
       {
        uzTekstas2 = "select distinct Pr_laikas from " + table2->TableName
                     + " where data = '" + data + "' and Pr_laikas <= '" + laikas + ":59' and Pb_laikas
>= '"
                     + laikas + ":00'";
        ApskaiciuotiSpreada(query,table2->DatabaseName,table2-
>TableName,laikas,uzTekstas2,spread,data);
```

```
  if(arVaizduoti)
   VaizduotiDuomenis2(seriesSp,spread);
 }
 if(arSandoriai)
 {
  uzTekstas = "select Laikass,Kaina,Kiekis from " + lenteleS
              + " where data = '" + data + "' and Laikas = '" + laikas + "' order by Laikass";
  SandoriuSarasas(query,keliasS,sandoriai,uzTekstas);
  NustatytiIniciatoriu(table2,sandoriai,data);
  for(int i = 0; i < (int) sandoriai.size(); i++)
   sandoriaiV.push_back(sandoriai[i]);
  if(arVaizduoti)
  {
   PiestiGrafikaSI(seriesSKa,sandoriaiV,true);
   PiestiGrafikaSI(seriesSKi,sandoriaiV,false);
  }

 }

}
//---------------------------------------------------------------------------
void TEfektyvumas::UzpildytiSandGridoA(TStringGrid*gridas)
{
 gridas->Cells[0][0] = "Akcija";
 gridas->Cells[1][0] = "Nuo";
 gridas->Cells[2][0] = "Iki";
 gridas->Cells[3][0] = "Dienų kiekis";
 gridas->Cells[4][0] = "Inic. pirkėjas, %";
 gridas->Cells[5][0] = "Inic. pardavėjas,%";
 gridas->Cells[6][0] = "Sand. kiekis";
 gridas->Cells[7][0] = "Sand. vertė, Lt";

}
//---------------------------------------------------------------------------
void TEfektyvumas::SandoriuInfo(TStringGrid*gridas,AnsiString akcija,AnsiString dataNuo,
                                AnsiString dataIki,int dienuSk,double inicPi, double inicPa,
                                int sandKiek,double apyvarta)
{
 int pglKiek;

 pglKiek = gridas->RowCount;

 gridas->Cells[0][pglKiek - 1] = akcija;
 gridas->Cells[1][pglKiek - 1] = dataNuo;
 gridas->Cells[2][pglKiek - 1] = dataIki;
 gridas->Cells[3][pglKiek - 1] = dienuSk;
 gridas->Cells[4][pglKiek - 1] = FloatToStrF(inicPi,ffFixed,8,2);
 gridas->Cells[5][pglKiek - 1] = FloatToStrF(inicPa,ffFixed,8,2);
 gridas->Cells[6][pglKiek - 1] = IntToStr(sandKiek);
 gridas->Cells[7][pglKiek - 1] = FloatToStrF(apyvarta,ffFixed,8,2);

 gridas->RowCount = pglKiek +1;
}
```