ŠIAULIŲ UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

INFORMATIKOS KATEDRA

Einaras Daujotas
Informatikos specialybės V kurso neakivaizdinio skyriaus studentas

ELEKTRONINĖS PARDUOTUVĖS PIRKĖJŲ IR LANKYTOJŲ VEIKSMŲ STATISTINĖS ANALIZĖS SISTEMA

THE SYSTEM OF STATISTICAL ANALYSIS OF ACTIONS OF E-SHOP CUSTOMERS AND VISITORS

BAKALAURO DARBAS

Darbo vadovas:
Lekt. M.Stoncelis

Recenzentė:
Lekt. L.Tankelevičienė

Šiauliai, 2011

„Tvirtinu, jog darbe pateikta medžiaga nėra plagijuota ir paruošta naudojant literatūros sąraše pateiktus informacinius šaltinius bei savo tyrimų duomenis“

Darbo autoriaus _____

(vardas, pavardė, parašas)

Darbo tikslai ir uždaviniai

Tikslas

Sukurti ir integruoti realiai veikiančioje elektroninėje parduotuvėje lankytojų ir pirkėjų veiksmų statistinės analizės sistemą, kuri potencialiai būtų naudinga parduotuvės administracijai siekiant didinti prekių pardavimus, bei panaudoti analizės rezultatus automatiškai generuojant pasiūlymus lankytojams.

Uždaviniai

- Ištirti veikiančią elektroninę parduotuvę, susipažinti su jos veikimu, valdymu bei programine dalimi. Esant būtinybei, atnaujinti sudėtines jos dalis.
- Sukurti ir integruoti elektroninėje parduotuvėje programinį modulį, fiksuojantį duomenis apie klientus. Rinkti tuos duomenis, kurie vėliau bus naudojami analizei.
- Sukurti ir integruoti elektroninėje parduotuvėje programinį modulį, kuris pateiks parduotuvės administracijai surinktus ir apdorotus duomenis.
- Sukurti keletą statistinių duomenų analizių numatant, kaip gauti rezultatai galėtų būti panaudoti plėtojant elektroninį verslą.
- Sukurti mechanizmą, kuris pasiremdamas apdorotais statistiniais duomenimis automatiškai pateiks parduotuvės lankytojui pasiūlymus dėl kitų prekių, susijusių su jo peržiūrima preke. Įvertinti lankytojų veiksmų pokyčius po jo įdiegimo.

Darbo vadovas _____
(vardas, pavardė, parašas)

Įvadas

Šiandien internetas yra tapęs universalia komunikacijos terpe, kurioje vis stipriau reikia kovoti dėl žmonių dėmesio, jei nori jiems pasiūlyti paslaugą. Siekiant patraukti dėmesį, kuo toliau, tuo įvairesnių priemonių ir būdų naudojama. Vienas iš ryškių kokybiškų paslaugų bruožų ir atitinkamai viena iš interneto paslaugų vystymo krypčių yra dirbtinio intelekto elementų naudojimas.

Žmogus – individuali būtybė, bet kartu ir grupinė. Tiriant individualių žmonių elgesį, stebint pasikartojimus, panaudojus statistikos metodus, galima sukurti jų elgesio modelius. Modelius, kurie tiks daugumai. Ir grįžtant atgal, tuos modelius pritaikyti atskiram individui, siekiant nuspėti jo tolimesnį elgesį.

Tokie metodai jau ne vienerius metus taikomi elektroninėje prekyboje. Daugelis didesnių elektroninių parduotuvių kaupia ir apdoroja lankytojų duomenis, tiria elgesį ir pagal tai priima verslo sprendimus. Įdiegia mechanizmus, kurie bando nuspėti kliento elgesį ir siekia tą elgesį nukreipti norima linkme. Elektroninė parduotuvė tampa intelektualiai aktyvi.

Be to reikia nepamiršti, kad elektroninės prekybos tikslas yra kuo daugiau parduoti. Parduodant naudinga pažinti savo klientą. Taigi bet kokia statistinė analizė ir tyrimas šiuo atveju neturi būti savitikslis – apie klientus ir jų veiksmus turi pateikti tokią informaciją, kuri bus reikšminga priimant verslo sprendimus. Atliekant analizę, turi būti tiksliai numatoma, ką gauta informacija duos verslo požiūriu, jei gaunami vieni rezultatai, ir ką – jei gaunami kiti.

Šio darbo tikslas ir yra paimti realiai veikiančią elektroninę parduotuvę, kurioje nėra pirkėjų ir lankytojų veiksmų statistinės analizės sistemos, ir joje sukurti tokią sistemą – paprastą, bet galinčią duoti naudingų rezultatų. Taip pat įdiegti sprendimus, paremtus statistine analize, kurių tikslas – įtakoti klientų veiksmus.

Elektroninė parduotuvė, kurioje visa tai bus įgyvendinama, prekiauja meno, dizaino gaminiais, išskirtine rankų darbo produkcija ir ją galima rasti adresu www.ona.lt ar www.ona.com. Konkreti parduotuvė iš karto uždeda savo apribojimus bei parodo veikimo kryptis, sukuria problemas, kurias reikia išspręsti. Parduotuvė veikia nuo 2004 m., tačiau parduodamų prekių netgi per visą tą laiką nėra tiek daug, kad galima būtų atlikti statistinę analizę, galinčią duoti daug naudos verslo požiūriu, kai nebuvo fiksuojama papildomų duomenų, o duomenys buvo fiksuojami tik tiek, kiek jų reikia sandoriui atlikti. Tie duomenys taip pat bus naudojami, tačiau pagrindinis dėmesys bus kreipiamas į išsamesnių duomenų surinkimą ir būtent šiais duomenimis paremtą analizę. Kadangi statistinei analizei visada geriau turėti kuo daugiau duomenų, reikia nepamiršti, kad įvertinant parduotuvės mastus, kai kurios analizės galės duoti labiau patikimos informacijos tik praėjus pusmečiui ar metams.

Teorinis pagrindimas

Elektroninės parduotuvės

Norint kurti ar modifikuoti elektroninę parduotuvę, reikia susipažinti su teoriniais tokių parduotuvių veikimo pagrindais. Norint keisti, tai netgi svarbiau nei norint sukurti. Nes sukurti vienaip ar kitaip galima vien sugalvojus, kaip tai turėtų atrodyti. Tačiau norint modifikuoti, supratimas apie tai, kaip elektroninės parduotuvės kuriamos ir kaip jos veikia, yra labai reikalingas. Vien skaitant programinį kodą užimtų daug laiko norint ištirti visą jo veikimą, todėl žinios apie galimus variantus labai praverstų.

Bene pirmas techninis klausimas, kuris sprendžiamas norint pradėti elektroninį verslą yra klausimas apie pasirinkimą – pasirinkti elektroninę parduotuvę kurti ant egzistojančių elektroninės komercijos karkasų (angl. framework) ar kurti savo sistemą. Yra daug kriterijų, kurie tokiu atveju galėtų nulemti pasirinkimą, tačiau iš programuotojo pusės svarbiausia yra tai, kad jau egzistuojančio karkaso panaudojimas reikalauja mažiau darbo programuojant, galima greičiau paleisti veikiantį produktą, tačiau reikia būti susipažinus su naudojama karkaso sistema. Iš kitos pusės savo sistemos kūrimas, reikalauja daugiau programavimo, tačiau leidžia sukurti nestandartinius sprendimus, taip pat leidžia lengviau ir laisviau plėsti sistemą.

Elektroninės komercijos sistemos yra interneto turinio valdymo sistemų (angl. CMS – Content Management System) rūšis arba išplėtimas. Tokių sistemų yra sukurta labai daug, pavyzdžiui, jų sąrašą ir savybes galima peržiūrėti ir palyginti tinklalapyje http://www.cmsmatrix.org/. Visų jų tikslas – palengvinti informacijos pateikimą internete. Jos sukurtos įvairiomis programavimo kalbomis, turi įvairias licencijas.

Iš egzistuojančių ir populiarių elektroninės komercijos sistemų (karkasų) galima paminėti, pvz., Drupal e-commerce, Magento, CubeCart. Pavyzdžiui, populiari atvirojo kodo sistema Drupal, kaip turinio valdymo sistema ir interneto aplikacijų (angl. web applications) karkasas, leidžia kurti įvairaus pobūdžio svetaines. O galimybė įdiegti naujus modulius labai išplečia šios sistemos panaudojimo ribas. Kadangi ši sistema yra populiari, galima būtų trumpai apibūdinti jos struktūrą kaip pavyzdį, siekiant suprasti elektroninės komercijos sistemos veikimo principus.

Taigi struktūra yra modulinė – sudaryta iš branduolio ir atskirų modulių. Branduolys atlieka pagrindines funkcijas: leidžia sukurti klasikinį tinklalapį su papildomomis funkcijomis. Tam tikslui yra sukurti komponentai su galimybe kurti ir valdyti turinį, failų įkėlimą, meniu, vartotojų paskyras, leidimus, katalogus, tinklaraščius, diskusijų forumus,

internetinius balsavimus. Branduolyje taip pat įdiegta ir galimybė administruoti svetainę per interneto naršyklę, įvairiapusis paieškos variklis, daugiakalbiškumo palaikymas ir kt.

Ir nors šių galimybių pakanka norint sukurti įvairiapusišką svetainę, bet dar yra tūkstančiai modulių, kuriais galima išplėsti sistemos galimybes norima linkme. Šių modulių yra tiek daug, nes Drupal yra populiari atvirojo kodo sistema parašyta PHP kalba, ir jos išplėtimo modulius gali kurti bet kas. Galima rasti beveik 8000 modulių, o vien elektroninei komercijai skirta virš 500 modulių [1].

Iš kitos pusės kad ir kokia plėtojama ir modeliuojama būtų sistema, ji gali netenkinti konkretaus elektroninio verslo. Tiriama parduotuvė www.ona.com iš dalies kaip tik tokia ir yra, todėl ji buvo sukurta individualaus karkaso struktūros naudojant PHP kalbą.

Kokias būtinas savybes individualus elektroninės parduotuvės karkasas paprastai turi? Pavyzdžiui, jos galėtų būti tokios [2]:

- Prekių talpinimas ir rodymas tinklalapyje;
- Prekių kategorijų talpinimas ir rodymas tinklalapyje;
- Prekių ir jų kategorijų įtraukimas ir naudojimas svetainėje kitais tikslais;
- Prekių savybių nustatymas, keitimas;
- Prekių paieška;
- Prekių filtravimas pagal kliento nustatytus kriterijus;
- Pageidavimų sąrašas (angl. wish list);
- Prekių siūlymų generavimas, paremtas kliento ankstesniais pirkimais;
- Elektroninių laiškų siuntimas klientams;
- Prekių apžvalgos ir įvertinimai;
- Prekių krepšelis;
- Siuntimo kainos skaičiavimas;
- Mokesčių išskaičiavimas;
- Nuolaidų kodai;
- Dovanų kuponai;
- Kita nuolaidų sistema;
- Mokėjimų už prekes apdorojimas;
- Klientų paskyros ir galimybė jiems patiems jas valdyti;
- Elektroninės parduotuvės administravimas.

Šiame sąraše nėra klientų analizės sistemos, kadangi ji nėra būtinas elektroninės parduotuvės elementas. Parduotuvė gali veikti ir be jos, kaip beje daug mažesnių elektroninių parduotuvių ir veikia. Tačiau paminėtas prekių siūlymų generavimas, paremtas kliento ankstesniais pirkimais, gali implikuoti tam tikrą duomenų analizę – bent jau to kliento pirkimų istorijos stebėjimą ir nustatymą, kokios prekės jam dar galėtų tikti.

Taip pat pažymėtina, kad galimi įvairūs elektroninių parduotuvių kūrimo būdai ir jų pagrindimai (kaip rodo ir daugelis jau sukurtų karkasų). Vienos parduotuvės kuriamos imant pagrindu nedidelį karkasą ir po to jį plėtojant įskiepiais, kitos kuriamos iš karto kaip sudėtingi

karkasai, turintys daug savybių. Programiniu požiūriu nurodyti elektroninės parduotuvės elementai gali būti realizuoti taip pat įvairiai.

Šiuo metu dažniausiai taikomas programų architektūros modelis yra: Modelis-Vaizdas-Valdiklis (angl. MVC - Model-View-Controller). Pagal šį modelį vaizdo pateiktis vartotojui (Vaizdas) yra atskirta nuo duomenų bei jų apdorojimo mechanizmo (Model) ir taip pat atskirtas įvykių fiksavimo bei perdavimo mechanizmas (Controller). Tokia architektūra labiausiai tinka internetui, nes leidžia sukurti daug Vaizdų ir Valdiklių komunikuojančių su tuo pačiu Modeliu. Čia Modelis gali apimti ir duomenų bazę, ir kitais būdais saugomus duomenis. Šie duomenys pateikiami Vaizdui ir atvaizduojami interneto naršyklėje. Vartotojui naršyklėje inicijavus veiksmus, pvz., pasiuntus paieškos užklausą, ši informacija perduodama Valdikliui, kuris atitinkamai gali pagal nustatytą programą perduoti užklausą Modeliui. Tokia architektūra leidžia kurti ir vystyti visas šias dalis nepriklausomai viena nuo kitos.

Elektroninės parduotuvės atveju šis modelis paprastai papildomas Registru (angl. Registry), kuriame saugomos bendro naudojimo funkcijos (struktūrinio programavimo atveju) ar klasės (objektinio programavimo atveju). Per Registrą paprastai pasiekiamos šios dalys:

- Šablonų mechanizmas;
- Elektroninių laiškų siuntėjas;
- Failų sistemos valdymas;
- Vartotojo autentifikavimo mechanizmas;
- Duomenų bazės valdymo mechanizmas.

Ir dažniausiai elektroninės parduotuvės failai diske saugomi atsižvelgiant į šią programos struktūrą, ją atkartoja. PHP kalboje paprastai pagrindinis failas *index.php*, į kurį kreipiasi vartotojo naršyklė, susieja visas šias programos dalis.

Interneto analizė aspektai

Naudinga žinoti, kas ir kaip naudoja jūsų svetainę, ypač jei kiekvienas naudotojas yra potencialus svetainėje pateiktų prekių pirkėjas. Nes norint kažką svetainėje keisti ir gerinti, reikia turėti duomenis ir juos išanalizuoti. Kadangi tai naudinga, tokią informaciją renka, kaupia ir analizuoja daugelis interneto svetainių. Čia galima išskirti kelis būdus, kuriais tai daroma:

- Naudojant savo duomenų fiksavimo mechanizmus ir kaupiant duomenis savo serveryje;
- Naudojant kitų gamintojų priemones ir kaupiant duomenis jų serveriuose, pvz. naudojant Google Analytics;
- Naudojant serverio įrašų (angl. logs) analizavimo programas, kad analizuoti savo serveryje sukauptus duomenis, pvz. Webalizer.

Pirmuoju atveju reikia sukurti duomenų fiksavimo mechanizmą, analizės priemones ir rezultatų pateikimą suinteresuotiems asmenims, ko daryti antruoju atveju nereikia, pvz., Google Analytics atveju užtenka savo tinklalapiuose įdiegti mažą pateikiamą JavaScript kodą, kuris surenka reikalingą informaciją apie naršymo aplinką, o jo iškviestas JavaScript failas ga.js (jame esantis kodas) iš GoogleAnalytics serverio, perduoda surinktą informaciją Google Analytics serveriui. Surinkti duomenys apibendrinami ir pateikiami Google Analytics tinklalapyje. Kaip matyti toks būdas yra žymiai paprastesnis ir nereikalauja iš programuotojo beveik jokių pastangų. Trečiasis būdas irgi nereikalauja pastangų, nes serveriai ir taip daro įrašus (ar bent jau gali daryti), reikia tik paleisti programą, kuri juos išanalizuos.

Iš kitos pusės paminėtas antrasis būdas, kaip beje ir trečiasis, turi ir pagrindinį trūkumą, visada būdingą tokiais atvejais, kai programa bando patenkinti daugelio vartotojų poreikius, – neatsižvelgiama į konkrečios svetainės specifiką. Pavyzdžiui, elektroninėje parduotuvėje juk ne visada aplankyti adresai parodys, kokia konkrečia preke domėtasi, ar kas tuo metu įdėta kliento prekių krepšelyje. Tai, kiek naudingos informacijos surinks išorinės priemonės labai priklauso nuo to, kaip bus suprogramuota svetainė. Be to ir duomenų analizės priemonių gali nepakakti konkrečiam vartotojui. Nekalbant apie tai, kad ir duomenų rinkimo mechanizmai gali nesuveikti ar suveikti, ne taip kaip turėtų. Tuo tarpu duomenis renkant, kai informacija fiksuojama tame pačiame serveryje, kuris pateikia svetainę, duomenys visada bus korektiškai surenkami, ir po to juos galima apdoroti, kokiais norima algoritmais ir pateikti bet kokiais norimais pjūviais. Be to jungti su bet kuriais kitais reikalingais duomenimis apie tą pačią svetainę (ir ne tik). O praktiniu požiūriu, norint gauti kuo daugiau duomenų, galima naudoti ir visus tris būdus. Ir jei dalį duomenų gali gauti išorinėmis priemonėmis, reikia jas ir naudoti, nes taip paprasčiau. O jei reikia sudėtingesnių duomenų apjungimų, specifinių duomenų, ar dar dažniau jų interpretacijų – tada sukurti ir savo priemones.

Taip pat svarbus klausimas yra surinktų duomenų interpretavimas. Kaip būdavo įprasta, vienais ar kitais būdais fiksuojant klientų paspaudimus ant tinklalapio nuorodų, skaičiuojant išbūtą laiką ar pan. gauti duomenys nebūtinai atsakys į norimus klausimus. Pavyzdžiui, keli iš tradiciškai fiksuojamų duomenų:

- Puslapių peržiūra. Ar daug peržiūrų yra geras rodiklis? Galbūt geras, nes reiškia, kad tinklalapyje yra daug naudingos informacijos. O galbūt blogas, nes yra nepatogi navigacija ir reikia eiti ratais, ar ilgą kelią kol randi naudingą informaciją. O gal trumpa navigacija reiškia, kad iš tikrųjų svetainė klientų nedomina.
- Kreipimaisi į serverį. Tradiciškai tai reikšdavo, kad daugiau turinio pateikiama klientui. Dabar dažnai vienas puslapis kreipiasi į serverį dešimtis ir daugiau kartų. Ką konkrečiu atveju turėtų reikšti tas kreipimųsi skaičius?

- Išėjimo puslapiai. Fiksuojami puslapiai, kuriuose lankytojas baigia naršymą. Ar tai reiškia, kad tas puslapis nepatinka lankytojui? Ar kad lankytojas rado tai, ko ieškojo ir tuo baigia? Gali būti ir taip ir taip.
- Svetainės naudojimas. Fiksuojant, unikalius lankytojus, bei sugrįžtančius lankytojus, gauti skaičiai taip pat gali būti nevienareikšmiški – ar lankytojas vis sugrįžta bandydamas rasti tai, ko nerado anksčiau, ar dėl to, kad svetainė labai įdomi?

Taigi duomenų interpretavimo klausimas šiuo atveju ne mažiau, o gal net labiau svarbus, nei pats tų duomenų turėjimas. Ir interpretavimas gali labai skirtis ir priklausyti nuo įvairiausių faktorių, pavyzdžiui, tokių, kaip svetainės paskirtis ir tikslas, tikslinių lankytojų auditorija ar netgi svetainės struktūra.

Elektroninės parduotuvės tikslas yra parduoti kuo daugiau prekių. Ir geras lankytojas yra tas, kuris perka, kad ir užėjęs vieną kartą, bet ne tas, kuris užėjęs dešimtis kartų nieko neperka. Ir, pavyzdžiui, toks kriterijus kaip užduoties pilnas įvykdymas, kuris kitokio pobūdžio svetainėse gali būti nelengvai nustatomas, elektroninėje parduotuvėje nustatomas nesunkiai – užduotis įvykdoma, kai atliekamas pirkimas.

Bet kokiu atveju tobuliausias duomenų rinkinys gali atsakyti tik į klausimą, kas įvyko. Tačiau, kad atsakyti į klausimą kodėl tai įvyko, reikia atlikti gerai apgalvotas analizes, ir net tada gavus rezultatus sužinosi tik dalį tiesos, galbūt labiau tikėtiną tiesos variantą.

Taigi čia išdėstytas trejetas: elgesio fiksavimas (kas?), pasiektas rezultatas (pirkimas), motyvacijos nustatymas (kodėl?) - yra platesnio požiūrio interneto svetainių analizėje įgyvendiaugnimas, nei tik paprastas duomenų rinkimas ir grafinis jų vaizdavimas [3].

Techniniu požiūriu galima išskirti tokias duomenų rinkimo priemones:

- serverio interneto įrašai (angl. web logs),
- JavaScript žymos,
- interneto „švyturiai“ (angl. web beacons) – minimalūs permatomi paveikslėliai, kuriuos siunčiant interneto naršyklei ar el. pašto programai su kitu turiniu tuo pačiu surenkami duomenys apie užklausėją;
- paketų tyrimas (angl. packet sniffing), kai visi tinklo paketai siunčiami per tarpinį programinį ar techninį įrenginį, kuris nuskaito tiklo paketo duomenis,
- svetainės kodą generuojanti programa – joje įdiegiamas papildomas kodas, atliekantis apskaitos veiksmus.

Šie būdai turi savų privalumų ir trūkumų, bei savo paskirties sferas. Pavyzdžiui, interneto „švyturiai“ patogūs tuo, kad integruoti HTML kode ir daugiausia naudojami reklaminėse kampanijose fiksuoti, ar buvo peržiūrėtas konkretus turinys. O interneto įrašai nepatogūs tuo, kad kartais galima neturėti galimybės prieiti prie serverio daromų įrašų, pavyzdžiui, jei interneto turinio talpintojas nesuteikia tokios galimybės. JavaScript naudojami, kai duomenys renkami trečios šalies programomis ir jos serveriuose, bet ne tik. Duomenys gali būti renkami ir savo JavaScript programomis ir įsigytais sprendimais, paremtais

JavaScript, kai duomenys saugomi savo serveryje (pavyzdžiui, ClickTracks). JavaScript naudingi, kai reikia surinkti kuo daugiau duomenų apie vartotoją, nes programa veikia jo kompiuteryje. Šiuo atveju JavaScript turi dar ir tokių privalumų, kaip galimybė apskaityti iš buferio (angl. cache) įkraunamus puslapius arba kaip skirtingos informacijos rinkimas skirtinguose puslapiuose. Šalia to, turi ir trūkumų – vartotojas gali būti išjungęs JavaScript palaikymą arba tokio palaikymo gali iš vis nebūti (pvz. telefonuose, naršančiuose internete), tada duomenys nebus renkami. Taip pat sunkiau yra rinkti duomenis apie atsisiuntimus (angl. downloads) arba nukreipimus (angl. redirect).

Paketų tyrimo metodas pasižymi tuo, kad gali surinkti tokių duomenų, kurių kitais būdais negausi, t.y. viską, ką įmanoma gauti nadojantis konkrečiu tinklo protokolu. Tačiau negalima bus surinkti duomenų, kurie neperduodami tinklu – tai duomenys apie vartotojo kompiuterį, ar kai kuriuos juo veiksmus (bet šiuos duomenis galima surinkti naudojant JavaScript). Taip pat tiriant paketus yra sunkiau konfigūruoti ir atsirinkti reikalingus duomenis, be to gali kilti problemų dėl privatumo, nes paketuose yra visa informacija, kurią siunčia vartotojas, tame tarpe ir konfideciali.

Visi šie būdai turi savo privalumų ir trūkumų, todėl visada sprendimą, kurį(-uos) variantą (-us) pasirinkti, reikia paremti konkrečių poreikių analize.

Projektinė dalis

Susipažinimas su elektronine parduotuve

Siekiant įdiegti elektroninės parduotuvės statistinės klientų analizės sistemą, praktinis darbas pradedamas nuo susipažinimo su elektronine parduotuve.

Vartotojo perspektyva

Pirmiausia susipažįstama iš vartotojo perspektyvos, užfiksuojant pagrindines savybes, kurios yra įdiegtos ir matomos parduotuvėje. Galima išvardyti tokias www.ona.com iš vartotojo perspektyvos pastebimas savybes:

- daugiakalbiškumo palaikymas (įdiegtos 3 kalbos: lietuvių, anglų, rusų),
- daugiavaliutiškumo palaikymas (įdiegtos 3 valiutos: LTL, EUR, USD),
- prekių krepšelis,
- vartotojo sąskaita, su galimybe klientui ją susikurti, įvesti norimus duomenis,
- saugaus protokolo palaikymas suvedant privačius duomenis,
- paprasta paieška (nekonfiguruojami užklausų parametrai),
- prekių katalogai,
- atsižvelgiant į menkas paieškos galimybes, naršymas nukreiptas naudoti prekių kategorijas ir per jas rasti norimą prekę,
- nedaug naudojami JavaScript,
- pageidavimų sąrašas,
- siuntimo kainų skaičiavimas,
- siuntimo įvairiais adresais galimybė,
- galimybė pasirinkti įvairių dydžių prekes,
- monogramų uždėjimas,
- pakuotės pasirinkimas,
- apmokėjimo variantų pasirinkimas.

Svetainės administratoriaus perspektyva

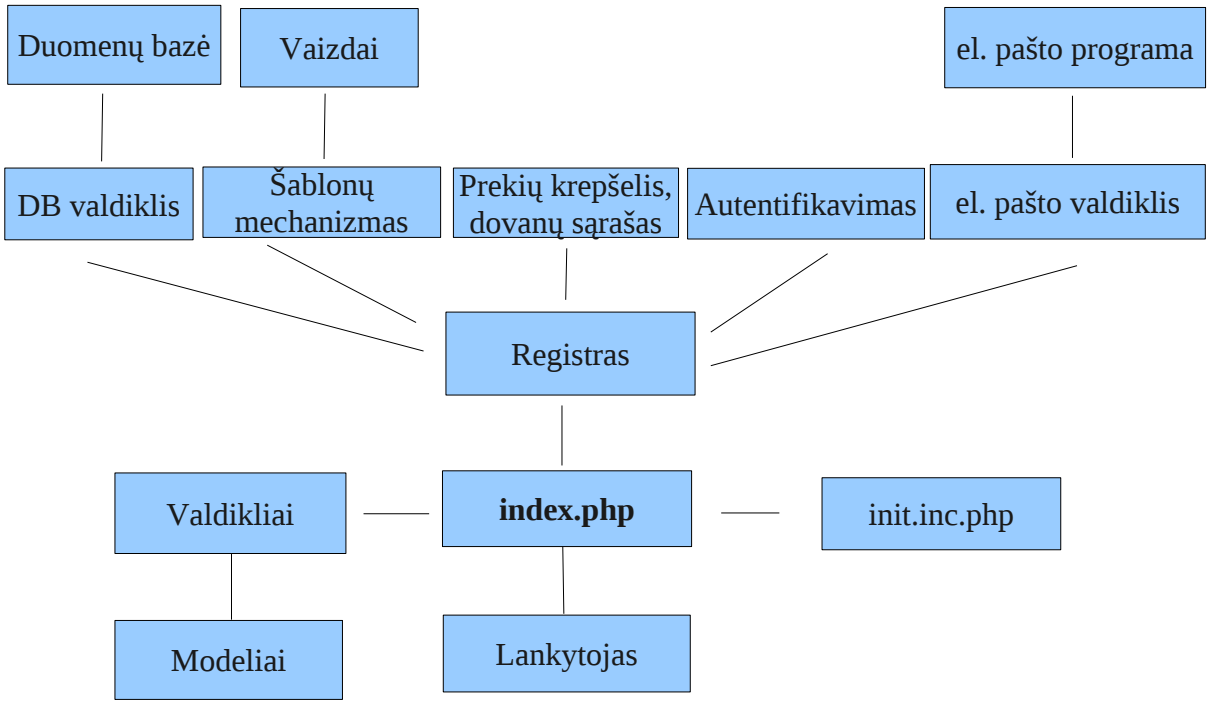
Taip pat būtinas susipažinimas su elektroninės parduotuvės administravimo sistema.

Administravimas atliekamas naršykle internetu prisijungus prie administravimo sistemos saugiu protokolu. Administravimo sistema sudaryta moduliniu principu, kiekvienas modulis atlieka savo funkcijas, yra galimybė modulius atjungti priklausomai nuo administratoriaus kategorijos. Yra tokie administravimo moduliai: struktūra ir turinys, pop-up langas, kalbos, darbų autoriai, prekių spalvos, valiutos, pristatymo kainos, titulinio puslapio nuotraukos, terminų žodynas, klientai, klientų kodai ir nuolaidos, užsakymai, ataskaitos, naujienlaiškiai, administratoriai, moduliai, nustatymai, DB atsarginės kopijos, administravimo vertimai.

Programuotojo perspektyva

Svetainė sukurta nadojant PHP programavimo kalbą ir MySQL duomenų bazių valdymo sistemą. Patalpinta serveryje, kuriame įdiegta PHP 4.4 versija ir MySQL 4.0 versija.

Susipažinus su svetainės failų struktūra bei pagrindinių failų kodu, galima būtų taip struktūriškai pavaizduoti svetainę pagal MVC modelį:



Pagrindinis svetainės failas, esantis ir registru, yra *index.php*, konfigūravimo nuostatos saugomos faile *init.inc.php*.

Iš šių bei kitų failų nustačius, kad svetainės duomenys saugomi vienoje MySQL duomenų bazėje pavadinimu ona_com, būtina susipažinti su šios duomenų bazės struktūra. Tai galima atlikti prisijungus prie prie serverio, kuriame saugoma duomenų bazė, panaudojus PHP programą PhpMyAdmin, skirtą būtent administruoti MySQL duomenų bazes per tinklą.

Duomenų bazę sudaro 35 lentelės, kurias ištyrus, matyti, kad duomenų bazėje kaupiama informacija apie prekes, sąskaitas, yra registruotų pirkėjų sąrašas bei įvairios pagalbinės lentelės. Duomenys tiek apie registruotų pirkėjų, tiek apie neregistruotų lankytojų veiksmus svetainėje nėra kaupiami.

Svetainė sukurta derinant struktūrinio ir objektinio programavimo paradigmas. Bendrai naudotas struktūrinis programavimas – taip sukurtas ryšys registras–valdiklis–modelis. Tačiau beveik visi bendro naudojimo mechanizmai sukurti naudojant atskiras klases:

- duomenų bazės valdymas vyksta naudojant ezSQL klasę (failas paimtas iš išorinių šaltinių)[4];
- Šablonai kuriami ir valdomi phemplate klase (failas iš išorinių šaltinių)[5];

- Prekių krepšelis kuriamas ir valdomas sCart klase (klasė sukurta elektroninės parduotuvės kūrėjų);
- Elektroninių laiškų siuntimą valdo PHPMailer klasė (paimta iš išorinių šaltinių)[6].

Svetainės administravimo sistema sukurta svetainės gamintojų, ji turi savo atskirą struktūrą su savo pagrindiniu *index.php* failu. Čia informacijos pateikimui nenaudojami šablonai, vaizdas komponuojamas kelių php failų pagalba: *header.inc.php*, *footer.inc.php* ir iš atitinkamo modulio *index.php* failo. Panaudotas struktūrinis programavimas.

Naudojami įrankiai

Svetainė sudaryta iš daug php bei kitų failų, ir su jais susipažinti galima prisijungus saugiu ryšiu prie serverio ir ten šiuos failus peržiūrint. Tačiau toks būdas yra nepatogus, ir susipažinimas gali užimti neproporcingai daug laiko atidarinėjant ir vis uždarinėjant failus, ieškant po atskirus failus išdėstytų funkcijų, klasių ar konfiguracinių nustatymų aprašymų, bei po to sugrįžinėjant į ankstesnius failus ir sekant programinio kodo veikimo logiką.

Tokioje situacijoje darbą žymiai palengvinti galėtų keletas programų.

Filezilla

Pirmiausia, norint, kad failai būtų patogiai ir greitai atidaromi, patogu juos būtų atidarinėti iš savo kompiuterio kietojo disko. Iš interneto serverio persikelti failus į vietinį diską patogiausia pasinaudojus ftp protokolu ir ftp ryšio programa. Galima panaudoti Filezilla programą, nemakamą, patogią ir veikiančią įvairiose operacinėse sistemose.

Eclipse ir PDT

Persikėlus failus galima juos peržiūrinėti ir redaguoti tekstinių failų peržiūros programomis su PHP sintaksės žymėjimu. Tokių programų yra gana daug, pavyzdžiui, linux operacinėje sistemoje (kurioje atliekamas šis darbas) galima pasinaudoti nemokamomis Bluefish editor, gPHPEdit ir kt. programomis. Tačiau dirbant su didesniai projektais patogiau yra naudoti integruotą programų kūrimo aplinką (angl. Integrated Development Environment – IDE). Pasirinkti galima vėlgi nemokamą, veikiančią įvairiose operacinėse sistemose ir galingą programą – Eclipse.

Eclipse iš pirmo žvilgsnio nėra paprasta ir lengvai įsisavinama programa, tačiau verta ją išmokti naudoti dėl kelių pagrindinių priežasčių:

- Ji sukurta jau daugiau kaip prieš dešimt metų, nuolat vystoma (kaip atvirojo kodo programa), turi stiprų programuotojų bendruomenės palaikymą, kas reiškia, kad jos branduolys yra gerai ištestuotas, užtikrina patikimą veikimą, ir,

tikėtina, jos vystymas nesustos, todėl laikas, sugaištas išmokti naudotis programa, nebus prarastas, ir programą dar ilgai galima bus naudoti;
- Eclipse yra sukurta su įskiepių (angl. plugin) palaikymu, šiais įskiepiais galima jai suteikti daug reikalingų naujų savybių, įdiegti įvairių programavimo kalbų palaikymą. Įskiepių įdiegimas ir atnaujinimas yra integruotas pačioje Eclipse;
- Plačios konfigūruojamumo galimybės ir pritaikomumas individualiems poreikiams;
- Eclipse yra nemokama, sukurta Java kalba ir veikia įvairiose operacinėse sistemose, kuriose bus įdiegta JRE (Java Runtime Environment).

Iš trūkumų galima paminėti, kad Eclipse yra daug kompiuterio resursų naudojanti programa, ir kuo daugiau įskiepių įdiegta, tuo daugiau resursų programa naudoja. Su ja patogu dirbti, jei bus pakankamai operatyviosios atminties, geras procesoriaus greitis ir nemažas monitoriaus ekranas.

Norint, kad Eclipse palaikytų PHP programavimą, galima įsidiegti keletą tam skirtų paketų. Šiuo metu galbūt daugiausia galimybių turi PDT (angl. PHP Development Tool) įskiepis. Jis suteiks galimybę redaguoti programos tekstą, automatiškai tikrinti jį, ieškoti klaidų (angl. debug).

Įdiegus Eclipse su PDT, galima susikurti naują PHP projektą, nurodant, kad į jį įeitų visi tiriamos svetainės failai. Be to patogi galimybė yra nurodyti PHP versiją, pagal kurią būtų vertinamas kodas, pvz. 4, 5.1 ar 5.3. Sukūrus projektą, Eclipse iš karto patikrina visus projekto failus ir parodo jų sintaksės klaidas, jei tokių randa. Programos teksto redaktoriuje nuolat gali tikrinti įvedamą kodą ir iš karto pažymėti vietas, kuriose randa programavimo sintaksės klaidų. Taip pat teikia variantus dėl galimų įvesti funkcijų pavadinimų ir parametrų.

Tiriant ne savo sukurtą svetainę daug padeda galimybė rasti failus ir vietas juose, kur dominantys kintamieji buvo apibrėžti, taip pat rasti funkcijas, kur jos buvo aprašytos. Eclipse suranda tarp projekto failų reikalingus failus ir parodo vietas, kur yra reikalingi duomenys. Jei tokios galimybės nebūtų, tektų sugaišti daug laiko atidarinėjant įvairius failus ir ieškant atitinkamų vietų juose. Taip pat patogiai galima atskirose panelėse matyti funkcijų ar klasių savybių arba metodų aprašymus.

Xdebug

Atliekant programavimo darbus su svetaine, esančia savo serveryje, t.y. tokia, prie kurios priėjimas nėra apribotas ir galima instaliuoti ką nori, labai naudingas įrankis yra klaidų padedančios ieškoti programos - „debugeriai“. Viena tokių – Xdebug. Tai taip pat atviro kodo programa, gana galinga ir labai naudinga programuojant PHP kalba, kuriai ji ir yra skirta.

Xdebug turi būti instaliuota ir veikti tame serveryje, kuriame yra ir svetainės PHP variklis. Kadangi Xdebug yra Zend (PHP variklio) plėtinys, ją galima konfigūruoti per php.ini

failą. Taip pat galima nustatyti, kad konfigūraciniai parametrai būtų skaitomi ir iš atskiro failo – xdebug.ini. Konfigūraciniame faile nustačius, kad Xdebug pasileistų kartu su PHP bei kad klaidos ar įspėjimai būtų išvedami į ekraną, jau galima naudotis Xdebug privalumais. O pasirinkus atitinkamas nuostatas [7], galima gauti daug norimos informacijos, kintančios vykdant PHP skriptą, – kintamųjų reikšmes, funkcijų parametrų reikšmes, objektų savybes. Be to galima nustatyti, kad informacija, išvedama į ekraną, būtų patogiai suformatuota.

Tačiau tikrasis Xdebug patogumas pasireišką šią programą naudojant kartu su Eclipse. Xdegub yra kliento – serverio struktūros, t.y. su klientu veikiančiu nutolusiame ar tame pačiame kompiuteryje, galima prisijungti prie Xdebug serverio, veikiančio su PHP varikliu. Tokia programos struktūra leidžia Eclipse prisijugti prie svetainės Xdebug serverio ir gauti iš jo visą reikalingą informaciją apie skripto vykdymą. Sukonfigūravus prisijungimą, Eclipse pateikia patogų interfeisą, kuriame galima sekti programos vykdymą žingsnis po žingsnio, stebėti visus reikalingus kintamuosius, nepasiklysti vykdomų failų hierarchijoje.

Apache, MySQL, kt.

Iš esmės aukščiau aprašytų įrankių užtenka, kad galima būtų patogiai redaguoti programinius failus ir juos talpinti svetainės serveryje. Bet, pvz., Xdebug nėra instaliuotas realios svetainės serveryje. Be to reikia atsižvelgti į tai, kad svetainė yra realiai veikianti ir reikia užtikrinti, kad ji atliekant pakeitimus veiktų be trukdžių. Todėl naudinga būtų programinio kodo veikimą pirmiausia pasitikrinti saugioje aplinkoje, t.y. savo kompiuteryje.

Tuo tikslu kompiuteryje galima įsidiegti atitinkamas programas, kurios veikia ir realiame svetainės serveryje: Apache interneto serverį su PHP (plius Xdebug palaikymu), MySQL duomenų bazių valdymo sistemos serverį bei programą patogiam MySQL duomenų bazės valdymui, pvz., tą patį kaip realiame serveryje phpMyAdmin arba MySQL Workbench.

Norint svetainės pakeitimus testuoti su duomenimis, kurie yra sukaupti svetainės MySQL duomenų bazėje, šiuos duomenis galima būtų eksportuoti iš realios svetainės (tokia funkcija yra numatyta phpMyAdmin programoje), o po to tuos duomenis importuoti savo kompiuteryje veikiančiame MySQL serveryje, sukuriant tokią pat duomenų bazę, su tomis pačiomis lentelėmis ir patogiausia – tokiu pat vardu.

Ir paskutinis dalykas, kurį reikia paminėti ir kurį reikėtų padaryti, norint, kad svetainė pilnai veiktų vietiniame kompiuteryje, yra teisingų failų kelių ir slaptažodžių nustatymas. Šios svetainės failuose tai galima padaryti keliais būdais; vienas iš jų – pakeisti faile *init.inc.php* keletą globalių konstantų (nustatytų su funkcija *define( )*), kurios nurodo pagrindinius prisijungimo kelius ir prisijungimo slaptažodžius. Aišku, reikia nepamiršti, kad šis pakeistas failas neturėtų būti perkeliamas atgal į svetainės serverį. Kadangi svetainė šiuo

požiūriu nebuvo suprojektuota ideliai, atsirado ir kitų vietų, kuriose buvo naudojami absoliutūs failų adresai, todėl teko ir juos pakeisti, jei programa kurioje nors vietoje neberasdavo tinkamo failo kelio.

Reikia pažymėti, kad dalį testavimo darbų reikėjo atlikti ir realiai veikiančioje svetainėje, nes taip patogiausia gauti realų užklausų srautą ir kontroliuoti, ar teisingai vykdoma programa. Tuo tikslu svetainėje buvo padarytas laikinas duomenų išvedimas į tekstinį failą ir taip stebima, kokie rezultatai gaunami. Kurti ir diegti programinį kodą realioje svetainės administravimo aplinkoje buvo paprasčiau, nes netinkamo veikimo galimybė čia mažiau pavojinga, kadangi ją galėtų pastebėti tik svetainės administratorius.

Svetainės atnaujinimas (paruošimas)

Svetainė buvo sukurta 2004 metais, todėl ir PHP bei MySQL versijos yra senos, daugelis taikytų sprendimų buvo geri tuo metu, tačiau dabar reikalaujantys atnaujinimo. Siekiant apskritai tobulinti svetainę ir nesusidurti su problema, kad norimos panaudoti funkcijos ar klasės reikalauja naujesnės PHP versijos, arba taip pat norint pasinaudoti daugiau galimybių turinčia MySQL serverio versija, nuspręsta svetainę perkelti į serverį su šiuolaikiškesne PHP 5.2 ir MySQL 5.0 versija. Perkėlimo metu taip pat naudinga išspręsti ir teksto kodavimo problemą – iki šiol duomenų bazėje įrašai buvo koduojami win-1257 koduote, kas keldavo problemų su ne lietuviškais ir ne angliškais simboliais. Daugiakalbėje svetainėje daugiabaičio kodavimo naudojimas, pvz. utf-8, būtų optimalus.

Perkeliant svetainę iškilo keletas ryškesnių problemų. Pirmiausia pažymėtina tai, kad svetainės kode daug informacijos kintamaisiais perduodama per GET ir POST užklausas, ir šie kintamieji naudojami kaip globalūs kintamieji tiesiogiai, o neimant jų reikšmių iš $_GET ir $_POST masyvų. Nuo PHP 5 versijos php.ini direktyva „Register globals“ standartiškai yra nustatyta „Off“ (kas padaryta dėl saugumo ir yra suprantama), o tai reiškia, kad kode kai kurie kintamieji nebegauna savo reikšmių ir programa, aišku, neveikia. Negalint pakeisti severio nustatymų, galima būtų emuliuoti buvusį veikimą visada visas GET ir POST perduodamas reikšmes priskiriant globaliems kintamiesiems, kas ir padaryta adminstravimo aplinkoje, prie kurios jungiasi saugiu protokolu tik registruoti vartotojai, - šiuo atveju panaudojant funkciją import_request_variables(). Tačiau viešai prieinamoje svetainės dalyje siekiant išlaikyti saugumo reikalavimą, teko peržiūrėti kodą ir reikalingose vietose atlikti tiesioginį reikšmių priskyrimus kintamiesiems iš $_GET ir $_POST masyvų.

Taip pat šiek tiek problemų sukėlė perėjimas ant unikodo (utf-8) – sukūrus naują duomenų bazę, kurios lentelėse tekstiniai simboliai saugomi utf-8 koduote, teko senosios duomenų bazės lenteles perkoduoti. Kadangi duomenų bazė nebuvo labai didelė tą galima

buvo padaryti išeksportavus lenteles į keletą failų, atsidarius juos tekstiniu redaktoriumi, pakeitus koduotės nustatymą ir išsaugojus utf-8 koduote. Vėliau šie failai importuoti į naująją duomenų bazę.

Dėl koduotės pakeitimo taip pat teko pakeisti kai kurias su teksto eilutėmis dirbančias funkcijas, pvz. tokias kaip strtoupper() ar strtolowerr() pakeisti jų analogais su mb_ priešdėliu, galinčias teisingai dirbti su daugiabaičiais simboliais.

Taip pat atlikta ir kitų, smulkesnių pakeitimų. Pavyzdžiui, pakeitus seną ezmysql klasę, skirtą supaprastinti užklausų siuntimui į duomenų bazę, naujesne jos versija, pastarojoje nebebuvo metodo dequote(), todėl kode reikėjo rasti vietas, kuriose jis naudojamas ir atsisakyti jo naudojimo dažniausiai jį pakeičiant escape() metodu.

Tyrimo eiga

Duomenų rinkimas

Pirmas žingsnis duomenų analizės procese yra tuos duomenis surinkti. Serverio surinkti įrašų duomenys (šioje svetainėje pateikiami Webalizer), taip pat Google Analytics surinkti ir pateikiami duomenys nors ir naudingi tam tikrais aspektais, tačiau neatsako į daugelį svarbių klausimų. Net ir atsakymą, į atrodo paprastą klausimą – kokios prekės dažniausiai peržiūrimos parduotuvėje – ne taip paprasta gauti. Ataskaitose net ir matant, kokios parduotuvės dalys dažniausiai lankomos, nėra lengva suprasti, kokios prekės peržiūrimos, nes svetainės adresai, susiję su prekėmis, formuojami tik skaičiais. Reikia įdėti papildomų pastangų, norint išsiaiškinti, kokia prekė iš tikrųjų peržiūrėta.

O sudėtingesnių lankytojų analizės klausimų, pavyzdžiui, siejant klientų apsilankymus su atliktais pirkimais, neturint savo surinktų duomenų išspręsti nepavyktų. Todėl pasirinkta svetainėje įdiegti savo duomenų rinkimo mechanizmą ir surinktus duomenis saugoti duomenų bazėje, kad bet kada galima būtų su jais atlikti bet kokią norimą analizę. Ir nuspręsta duomenis rinkti php skripto pagalba serveryje, neperduodant šio darbo kliento naršyklei (kaip būtų daroma JavaScript atveju).

Visi navigacijos svetainėje veiksmai atliekami per pagrindiniame kataloge esantį *index.php* failą, t.y. serveris užklausą į svetainę perduoda šiam failui (kaip nustatyta serverio *.htaccess* faile). Užklausos įkelti paveikslėlius, ar svetainės tinklalapiuose esančių JavaScript'ų veiksmai, kurie taip pat tik įkelia paveikslėlius, nefiksuotini, nes reikšmingų klientų veiksmų jie neatspindi, o jų fiksavimas tik bereikalingai padidintų duomenų įrašomų į duomenų bazę srautą. Duomenys bus rašomi į duomenų bazę, o ne į tekstinius failus, nes dirbti su duomenų bazėmis yra patogiau, nei duomenis filtruoti iš tekstinių failų.

Taigi esant tokiai situacijai, duomenų rinkimas ir fiksavimas turėtų būti atliekamas vykdant *index.php* failą. Duomenų rinkimo kodą galima įrašyti tiesiai šiame faile arba kitame faile ir pasinaudojus *include_once()* funkcija įtraukti tą naująjį failą vykdant *index.php*. Siekiant įdiegiamą sistemą padaryti autonomiškesne, pasirinktas antrasis variantas ir sukurtas failas *analysis.php* (failas pridedamas).

Kuriant duomenų rinkimo mechanizmą šiuo atveju reikia išspręsti keletą problemų.

Svetainėje įgyvendintas php prisijungimo sesijų mechanizmas, kuris pagal serverio nustatymus šiuo atveju realizuojamas per slapukus (angl. cookies). Sesijos mechanizmas sukuria kliento naršyklėje savo slapuką, kuris galioja sesijos metu, ir pagal jį identifikuojama, kad naršo tas pats lankytojas. Tačiau svetainė nesukuria ilgai išliekančio slapuko, kuris leistų identifikuoti, kada sugrįžta jau anksčiau buvęs lankytojas. O tokia informacija gali būti reikalinga ir naudinga. Todėl svetainė turėtų sukurti savo ilgiau nei sesija galiojantį slapuką. Taigi sukuriamas toks slapukas panaudojant funkciją *setcookie()*, slapukas generuojamas jungiant lankytojo ip adreso dalis su atsitiktiniu skaičiumi, slapuko galiojimas nustatomas 2 metams, kiekvieno apsilankymo metu galiojimą pratęsiant.

Kita problema yra paieškos robotai. Kadangi šie „lankytojai“ nesusiję su galimais pirkėjais, o siunčia į svetainę gana daug užklausų, tai juos reikėtų išmesti iš fiksuojamų lankytojų sąrašo ir neapkrauti duomenų bazės įrašais apie juos. Tai padaryti galima pasinaudojus faktu, kad paieškos robotai nepriima slapukų, todėl patikrinus, ar lankytojas turi sesijos slapuką, galima išfiltruoti visus paieškos robotus.

Čia reikia pažymėti, kad pirmojo kreipimosi metu kliento naršyklė dar neturės sesijos slapuko, o nauja sesija bus sukuriama jau po užklausos gavimo, todėl norint neprarasti pirmojo kreipimosi duomenų to kreipimosi duomenis galima surašyti į sukurtą sesijos masyvą $_SESSION, o antrojo kreipimosi metu, kai jau lankytojas turės sesijos slapuką, juos iš sesijos masyvo įrašyti į duomenų bazę. Tiesa, filtruojant pagal slapukų palaikymą, iš apskaitos iškrenta naršyklės, kurios nepriima slapukų ir nepalaiko per juos sesijų, bet tai nedidelė problema, nes tokių atvejų nėra daug, o svarbiausia – šie lankytojai vis tiek negalėtų būti pirkėjais, nes tam būtinas užsiregistravimas ir sesijų slapukų palaikymas. Be to mechanizmas, kai duomenys į duomenų bazę įrašomi tik iš antro kreipimosi, išfiltruos atsitiktinai čia užklydusius lankytojus, kurie svetaine nesidomi ir joje nebando naršyti.

Dar būtų naudinga neapkrauti duomenų bazės (ir neapsunkinti vėlesnio duomenų filtravimo) įrašais apie svetainės administratorių veiksmus svetainėje. Administruojantys asmenys dažnai naršo svetainėje, stebi, kaip atrodo prekės, keičia, peržiūrinėja; kartais tam tikslui naudoja skirtingus kompiuterius ir skirtingas naršykles. Todėl reikia sąrašo naršyklių, kurios būtų nefiksuojamos duomenų bazėje. Tam tikslui duomenų bazėje sukuriama

papildoma lentelė ona_visitors_exclude sudaryta iš dviejų stulpelių – id ir visitors_id (kas yra sukurtas nuolatinis naršyklės slapukas). Joje bus įrašytos visos naršyklės, apie kurių veiksmus svetainėje informacija renkama nebus. Kadangi duomenys šioje lentelėje gali kisti, reikia sudaryti galimybę administratoriui pačiam paprastai save įrašyti į lentelę ir ištrinti iš kitų lentelių jau anksčiau buvusius jo įrašus. Tai įdiegta lankytojų apskaitos modulio faile *index.php* (pridedamas, išsamiau bus aprašytas kitame skyriuje).

Ir prie visų išvardytų nefiksuojamų duomenų, tuo pačiu *if* išfiltruojamas ir naršyklės atskiras kreipimasis į svetainės ikonos failą *favicon.ico*, nes tų kreipimųsi yra daug ir nėra prasmės duomenis apie juos saugoti duomenų bazėje.

Dar vienas spręstinas klausimas – kokius duomenis fiksuoti ir kaupti? Šių duomenų neturi būti per daug, kad neapkrauti bereikalingais duomenimis duomenų bazės serverio, bet turi būti tie duomenys, kurie gali būti reikšmingi darant išvadas apie klientų elgesį. Reikėtų fiksuoti užklausos laiką, serverio slapuko ir sesijos slapuko duomenis, kuriuos galima paimti iš automatiškai sukuriamo superglobalaus kintamojo (masyvo) $_COOKIE, taip pat užklausos adresą, kurį galima paimti taip pat iš automatiškai sukuriamo masyvo $_SERVER, su raktu $_SERVER['REQUEST_URI']. Taip pat naudingi gali būti ir kai kurie kiti duomenys iš $_SERVER masyvo: $_SERVER['HTTP_USER_AGENT'] parodantis duomenis apie lankytojo naršyklę ir operacinę sistemą, $_SERVER['REMOTE_ADDR'] parodantis lankytojo IP adresą, $_SERVER['REFERER'] parodantis, iš kokio tinklalapio naršyklė atėjo. Taip pat gali būti naudingi papildomi duomenys apie serverį, iš kurio naršo lankytojas, t.y. jo pavadinimas, gaunamas panaudojus funkciją *gethostbyaddr()*. Kiti naudingi duomenys yra apie lankytojo būklę, kiek ji susijusi su lankoma svetaine. Todėl duomenų bazėje fiksuojama tai, kokią valiutą pasirinkęs lankytojas (kokią kalbą jis pasirinkęs galima gauti iš užklausos adreso); koks yra kliento numeris, jei tas lankytojas naršo kaip registruotas svetainėje asmuo; kas yra kliento prekių krepšelyje. Atskirai pažymėtina, kad svetainėje atliekamų paieškų duomenys duomenų bazėje nekaupiami, kadangi užklausa programai perduodama su GET, kas reiškia, kad ji atsispindi $_SERVER['REQUEST_URI'] kintamajame.

Kadangi svetainė naudoja ir GoogleAnalytics, tai reiškia, kad šis JavaScript'as sukuria lankytojo naršyklėje papildomų slapukų [21], pagal kuriuos GoogleAnalytics renka statistinius duomenis. Naudingi šiuo atveju galėtų būti slapukai __utma ir __utmz, kurie be to, kad skirti lankytojui identifikuoti, savo pavadinime inkorporuoja ir papildomos informacijos apie lankytojo naršymą. Siekiant gauti daugiau informacijos, galima būtų ją išskaityti iš šių slapukų. Todėl juos taip pat naudinga įrašyti į duomenų bazę kartu su kitais duomenimis.

Sprendžiant, kuriuos duomenis fiksuoti, prieš sukuriant duomenų bazės lenteles, buvo naudinga nurodytą informaciją testavimo tikslais rinkti realioje svetainėje ir rašyti į

tekstinį failą, pvz., panaudojant funkciją *var_dump()*. Panaudojus šią funkciją aiškiai galima pamatyti, kokie duomenys yra saugomi atskiruose kintamuosiuose, kokia tų kintamųjų struktūra, koks duomenų ilgis (kas reikalinga kuriant duomenų bazės laukelius). Keletą dienų kaupiant atitinkamus duomenis įrašų failuose, po to galima peržiūrėjus nuspręsti, kuriuos duomenis iš tikrųjų naudinga įrašyti į duomenų bazę.

Kuriant duomenų bazės lenteles pagal nurodytus duomenis iš karto aišku, kad turi būti bent dvi lentelės. Vienoje lentelėje galima saugoti visus duomenis išskyrus tai, kokios prekės yra naršančio kliento krepšelyje. Kadangi tokių prekių gali būti įvairus kiekis, geriausia duomenis apie jas saugoti atskiroje lentelėje (pavadintoje *visitors_items*) su pagrindine lentele (pavadinta *visitors*) ją jungiant per svetimą raktą, kuris yra pagrindinės lentelės pirminis raktas (*id*, užklausos įrašo numeris, automatiškai didinamas vienetu).

Dar pažymėtina, kad visus naršymo duomenis saugant vienoje *visitors* lentelėje ši reliacinė duomenų bazė nėra pilnai normalizuota, nes yra pasikartojančių duomenų eilutėse (tokių kaip sesijos ar nuolatinio slapuko įrašai). Tačiau šis trūkumas nėra esminis ir turi įtakos tik duomenų bazės dydžiui. Bet kadangi tų pasikartojimų santykinai nebus daug ir atitinkamai vietos sutaupoma taip pat nedaug, o jų neiškeliant į atskiras lenteles užklausos duomenų bazei būtų paprastesnės, tai pasirinkta juos saugoti vienoje lentelėje. Juo labiau, kad jei ateity būtų poreikis, bet kada galima būtų atlikti padalijimą (lentelių struktūra pateikiama prieduose).

Tai yra pagrindiniai klausimai, kuriuos reikėjo išspręsti, o juos išsprendus belieka parašyti trumpą kodą php kalba, suformuoti SQL užklausas ir reikiamus duomenis siųsti į duomenų bazę. Po to galima prisijungti prie duomenų bazės su phpMyAdmin ir stebėti, kaip kuriami įrašai apie lankytojus.

Administravimo modulio sukūrimas

Surinkus duomenis, reikia juos pateikti svetainės administratoriui-vadybininkui. Duomenys turėtų būti pateikiami administravimo aplinkoje. Kadangi ši aplinka yra modulinės struktūros, patogiausia bus sukurti naują modulį ir padaryti atitinkamą naują meniu punktą.

Administravimo aplinkoje nėra įdiegtos galimybės automatiškai kurti naujus modulius. Ištyrus programinį kodą, matyti, kad pagrindiniai duomenys apie modulius imami iš MySQL duomenų bazės lentelės, skirtos šių modulių aprašymui saugoti. Paėmus iš lentelės pavadinimą bei kitus parametrus, php kodas suformuoja atitinkamą administravimo aplinkos meniu punktą. Taigi suvedus naujojo modulio *„Visitors_reports“* duomenis, toks punktas jau suformuojamas administravimo meniu.

Taip pat iš pagrindinio administravimo failo *index.php* kodo bei failų struktūros matyti, kad modulio programinis kodas imamas iš tam moduliui skirto katalogo ir jame

esančių failų *index.php* ir *actions.php*. Todėl toks katalogas *mod_visitors* bei failai sukuriami, ir juose turės būti reikalingas kodas.

Reikia išspręsti klausimą, kaip ataskaitų rezultatai bus pateikiami. Kadangi ataskaitų skaičius nėra iš karto nustatytas, o ir iš karto nustatyti būtų neproduktyvu, nes visada gali atsirasti poreikis gauti duomenų naujais pjūviais, tai aišku, kad galimų ataskaitų sąrašas turėtų turėti galimybę būti nesunkiai plečiamas. Taigi meniu punktas *„Ataskaitos apie lankytojus“* turėtų pateikti galimybę pasirinkti norimą ataskaitą. Todėl pagrindiniame modulio faile *index.php* (pridedamas) sukuriamos tik nuorodos į konkrečias ataskaitas ir įrašomas kodas, kuris pasirinks, kurios konkrečios ataskaitos kodą vykdyti. O tas kodas kiekvienai konkrečiai ataskaitai bus sukuriamas ir įrašytas jai skirtame faile, talpinamame tame pačiame kataloge. Tokiu būdu, norint įdiegti papildomą ataskaitą, užteks failą *index.php* papildyti viena nauja eilute su nuroda ir sukurti naują atitinkamai ataskaitai skirtą php failą. *index.php* faile, kaip nesusijusiame su konkrečia ataskaita įdiegtas ir kodas, kuris vienu mygtuko paspaudimu realizuos jau minėtą administratoriaus naršyklės įrašymą į nefiksuojamų naršyklių lentelę bei jau užfiksuotų duomenų ištrynimą iš duomenų bazės lentelių *visitors* ir *visitors_items*.

Kadangi administravimo aplinkoje taip pat įgyvendintas daugiakalbiškumas, tai kitas svarbus klausimas buvo išsiaiškinti, kaip jis yra įgyvendintas. Jei svetainės aplinkoje daugiakalbiškumas įgyvendintas visų naudojamų kalbų atitikmenis įvedant duomenų bazės lentelėje, tai administravimo aplinkos vertimai įgyvendinami kitaip. Čia yra sukurta funkcija *WA()*, kuri pateikiamam žodžiui, suranda atitikmenį reikalingoje kalboje, o šie atitikmenys yra saugomi tekstiniame faile ir įkraunami į duomenų masyvą, atitinkamai pagal naudojamą modulį. Kiekvienas modulis turi savo tokius vertimų atitikmenų tekstinius failus. O žodžių atitikmenų įvedimui yra padarytas atskiras administravimo aplinkos modulis *„Administravimo vertimai“*. Taigi norint ataskaitose išvedinėti į ekraną kokį nors tekstą, laikantis veikiančios sistemos principų, tekstas turėtų būti pateikiamas per funkciją *WA()* pateikiant šiai funkcijai teksto anglišką atitikmenį ir po to jau administravimo aplinkoje įvedant to teksto vertimą į norimą kalbą.

Nurodytų atlikti veiksmų pakanka, kad klientų duomenų analizių ir ataskaitų modulis būtų įdiegtas administravimo aplinkoje. Lieka sukurti pačias duomenų analizes ir ataskaitas.

Duomenų analizės ir ataskaitos

Realių lankytojų duomenų kaupimas pradėtas nuo 2011-01-08. Kol duomenų yra nedaug, jų statistinė analizė taip pat yra ribota. Todėl pradžioje sukurtos paprastesnės, bet administratoriui naudingos duomenų pateiktys: apie prekes, kuriomis labiausiai domimasi, apie prekes, kurias matė daugiausia žmonių, apie tai, kiek „tikrų“ lankytojų naršė svetainėje.

Šioms pateiktims sukurtas bendras interfeisas (failas *dates_chooser.inc.php*), kurį įjungus į ataskaitos failą, atsiranda galimybė pasirinkti datas „nuo–iki“ laikotarpio, už kurį norima gauti ataskaitą, bei atskiros nuorodos greitesniam duomenų gavimui už šiandien, vakar, už savaitę, mėnesį ir metus. Taip pat padaryta galimybė pasirinkti, kiek įrašų išvesti į ekraną. Numatytosios reikšmės nustatytos tokios, kad duomenys patogiausiai tilptų į ekraną jo neperstumiant.

Kuriant duomenų analizes ir ataskaitas buvo išbandyti skirtingi PHP ir MySQL sąveikos principai. Esant nedaug duomenų, buvo naudotas principas, kai paprastu SELECT sakiniu perkeliami visi reikalingi duomenys iš duomenų bazės, o po to PHP priemonėmis tie duomenys atrenkami, rūšiuojami, išvedami, t.y. visas skaičiavimo darbas pavestas PHP skriptui. Toks metodas naudingas tuo, kad kiekviename duomenų atrinkimo etape galima stebėti jų kitimą, pasitikrinti, ar iš tikrųjų pasiekiamas norimas rezultas (tokiu principu parašytas kodas palikęs kaip pavyzdys komentaruose atitinkamuose php failuose).

Tačiau šiuo principu sukurtą kodą tenka pakeisti, kai reikia apdoroti didesnius duomenų kiekius. Čia racionaliau, o esant dar didesniam duomenų kiekiui – netgi būtina (nes duomenų kiekio tarp PHP ir MySQL perdavimas yra gana ribotas), pagrindinį duomenų apdorojimą perduoti MySQL serveriui, suformuojant sudėtingesnes SQL užklausas, o PHP varikliui palikti galutinį jų sutvarkymą ir pateikimą.

Be minėtų trijų ataskaitų, vėliau sukurtos dar trys analizės, kurios naudingesnės turėdamos kuo didesnį duomenų kiekį: prekių, kuriomis domimasi, bet jos neperkamos, prekių, kuriomis nesidomima, ir prekių, kurios parduodamos efektyviausiai.

Prekės, kuriomis labiausiai domimasi

Kaip buvo minėta administratoriui nėra patogu pasinaudojus Google Analytics ar Webalizer aiškiai matyti, kurios prekės susilaukia didžiausio susidomėjimo iš lankytojų. Turint noro, galima ir pagal šias ataskaitas susirasti reikalingų duomenų, bet jos nėra pateikiamos patogia forma. O tokių duomenų turėjimas yra naudingas norint žinoti į kurių prekių užsakymą ar vystymą atkreipti daugiau dėmesio. Todėl pirmiausia sukurta paprasta, bet galinti būti naudinga ataskaita apie prekes, kuriomis lankytojai labiausiai domisi (pridedamas failas *interested.inc.php*).

Prekių atrinkimas ir surūšiavimas šiuo atveju paremtas mintimi, kad prekių peržiūrų skaičius rodo susidomėjimą jomis. To paties asmens kelis kartus peržiūrėta prekė skaičiuojama kaip ir kito asmens peržiūrėta, laikant, kad realaus domėjimosi prasme pakartotinis peržiūrėjimas yra nemažiau svarbus už atskirą.

Siekiant iš duomenų bazės išgauti įrašus apie konkrečių prekių peržiūrėjimus, reikia pirmiausia iš adreso, saugomo laukelyje REQUEST_URI, išskirti dalį su prekės kodo įrašu. Tai galima padaryti, nes peržiūrint konkrečią prekę šis įrašas suformuojamas adrese stabilioje vietoje nuo pradžios, pvz. /lt/products/279/2021 arba /en/products/265/2021/2/, čia prekė 2021. Tik reikia atmesti tokios formos panašius adresus: /lt/products/279/page/1/. Tam tikslui galima panaudoti MySQL operatorių REGEXP su nustatytu šablonu bei funkciją SUBSTRING_INDEX(), atrenkančią dalį eilutės pagal nurodytą skirtuką. Šis principas bus naudojamas ir kitoms ataskaitoms, kai reikės išfiltruoti aplankytas prekes. Tuo pačiu sugrupavus ir susumavus, užklausa atrodytų taip:

```
$sql = "SELECT SUBSTRING_INDEX(SUBSTRING_INDEX(request_uri,'/',5),'/',-1) AS product, COUNT(*) AS count FROM ".PREFIX."_visitors WHERE time BETWEEN '".strftime("%Y-%m-%d %H:%M:%S", $f_since_stamp)."' AND '".strftime("%Y-%m-%d %H:%M:%S", $f_till_stamp)."' AND request_uri REGEXP '^/[a-z]{2}/products/[0-9]+/[0-9]+($|/.*)' GROUP BY product";
```

Šiek tiek apdirbus gautą rezultatą, jį galima išvesti į ekraną. Ir kas šiuo atveju patogiausia administratoriui (įgyvendinta ir kitose ataskaitose), tai išvedant duomenis pagal prekių numerius, galima įterpti naudingos informacijos apie prekę, kad ta informacija būtų lengvai suvokiama – pirmiausia pateikti prekės pavadinimą, aprašymą, taip pat galima išvesti prekių likutį, kainą ir kitus papildomus duomenis, kuriuos galima gauti iš duomenų bazės lentelių, skirtų prekių aprašymui.

Prekės, kurias matė daugiausia žmonių

Šios duomenų pateikties tikslas – parodyti, kiek unikalių lankytojų matė konkrečias prekes (pridedamas failas *interested_by_unique.inc.php*). Duomenų atrankos principas panašus į prieš tai buvusį, tik čia dar papildomai reikia rezultatus sugrupuoti pagal sesijos slapuką ona_uid. Ir jei prieš tai dar buvo palikta šiek tiek duomenų rūšiavimo darbo ir PHP varikliui, tai šiuo atveju visi reikalingi duomenys gaunami suformavus sudėtingesnę SQL SELECT užklausą (dėl aiškesnio užrašymo užklausos eilutę padalijus tarp kelių kintamųjų):

```
$subsql="SELECT SUBSTRING_INDEX(SUBSTRING_INDEX(request_uri,'/',5),'/',-1) AS product, statid FROM ".PREFIX."_visitors WHERE time BETWEEN '".strftime("%Y-%m-%d %H:%M:%S", $f_since_stamp)."' AND '".strftime("%Y-%m-%d %H:%M:%S", $f_till_stamp)."' AND request_uri REGEXP '^/[a-z]{2}/products/[0-9]+/[0-9]+($|/.*)' GROUP BY product, statid";
$sql = "SELECT product, COUNT(*) AS counter FROM (".$subsql.") AS tt GROUP BY product ORDER BY counter DESC";
```

Belieka tik atlikti gautų duomenų pateikimą norima forma.

Unikalūs lankytojai

Siekiant sužinoti, kiek svetainėje lankėsį „tikrų“ lankytojų, t.y. ne paieškos robotų, ne administratorių, o tų potencialių klientų, kurie čia atliko bent vieną naršymo judesį (ne tik įėjo ir išėjo), *visitors* lentelėje sukaupti duomenys leidžia lengvai tai padaryti (pridedamas

failas *unique_visitors.inc.php*). Be to nesunkiai galima nustatyti, kurie iš lankytojų užsiregistravo, kokie jų duomenys yra įvesti duomenų bazėje (ko apskritai nepadarysi su „išorinėmis“ analizės priemonėmis).

Šiuo atveju SQL užklausos paprastos, pasinaudojus SELECT DISTINCT user_id atrenkant registruotus lankytojus, o SELECT COUNT(DISTINCT statid) skaičiuojant, kiek buvo unikalių lankytojų.

Be to čia realizuotas ir lankytojų vidurkio per dieną norimam laikotarpiui skaičiavimas.

Prekės, kuriomis niekas nesidomi

Informacija apie produktus, kurių niekas neperžiūrinėja ar peržiūrinėja retai, taip pat naudinga parduotuvės administratoriui. Sukuriant šių prekių atrankos ir pateikimo mechanizmą, panaudotas šiek tiek kitas principas, mažiau apkraunantis MySQL serverį ir labiau – PHP variklį (pridedamas failas *notinterested.inc.php*). Čia sukurta atskira funkcija, padaranti peržiūrėtų per pateikiamą laiko tarpą prekių sąrašą, - get_viewed_products_array(), ir ši funkcija įdėta į bendrą šio modulio failą *actions.php* (pridedamas). Atskira funkcija sukurta numatant ją panaudoti ir atliekant kitas analizes. Joje SELECT užklausa paprastesnė naudojant ne REGEXP, bet LIKE operatorių:

```
$sql ="SELECT DISTINCT request_uri FROM ".PREFIX."_visitors WHERE time BETWEEN
    '".strftime("%Y-%m-%d %H:%M:%S", $since)."' AND '".strftime("%Y-%m-%d %H:
%M:%S", $till)."' AND request_uri LIKE '/__/products/_%/_%'"
```

O galutinis prekės kodas išskiriamas panaudojus PHP funkciją preg_match su šablonu '{^/../products/\d+/(\d+)}': preg_match('{^/../products/\d+/(\d+)}', $col, $match)

Įdomu, kad ši funkcija naudoja Perl standarto šablono formavimą, kai tuo tarpu MySQL REGEXP operatorius naudoja POSIX standartą, todėl jų sintaksė skiriasi. Šiuo atveju taip pat racionaliau būtų naudoti anksčiau aprašytą SELECT su REGEXP, bet turėti šabloną prekės kodui išskirti PHP funkcijoje taip pat gali būti naudinga.

O neperžiūrimų prekių sąrašas formuojamas palyginant visas peržiūrėtas prekes su visomis parduotuvėje matomomis prekėmis, panaudojant funkciją array_diff(), išskiriančią vieno masyvo įrašus, nesančius kitame masyve.

Prekės, kuriomis daug domimasi, bet jos neperkamos

Šios analizės tikslas – išskirti problematiškas prekes (pridedamas failas *problematic.inc.php*). Problematiškas ta prasme, kad jos kelia didžiausią susidomėjimą lankytojų tarpe, tačiau jų parduodama neproporcingai šiam susidomėjimui mažai arba iš vis

neparduodama. Tokia informacija administratoriui duotų pagrindą pagalvoti, ką reikėtų keisti šiose prekėse (pavyzdžiui, keisti kainą ar įtraukti prekę į pardavimo akcijas), kad susidomėjimas preke pavirstų į pardavimą. O pritaikius vienas ar kitas priemones, stebėti ar prekė pasitraukė iš problematiškųjų sąrašo.

Problematiškumą čia atspindėtų santykis tarp pirkimų ir prekės peržiūrėjimų – tai būtų prekės pardavimo tikimybė prekių peržiūrų erdvėje. Bet vartojui suvokti paprastesnis būtų priešingas santykis – jis rodytų, kiek kartų prekė peržiūrima, kol nuperkama. Pavadinkime šį santykį prekės pardavimo (neefektyvumo) reitingu.

Skaičiuojant reitingą reikia išspręsti klausimą dėl atvejų, kai neparduota nei viena prekė – tokiu atveju matematiškai būtų dalyba iš nulio, bet šią problemą galima būtų nesunkiai išspręsti. Svarbiausia yra duomenų prasmė. Ir čia jau reikia atsižvelgti į prekybos specifiką. Kadangi prekiaujama mažo tiražo gaminiais, dažnai ir vienetiniais, tai skirtumas tarp vieno pardavimo ir nė vieno yra esminis – vieno prekės pardavimo negalima vertinti kaip vieno iš daugelio tos rūšies prekių pardavimo. Čia vienas pardavimas gali reikšti 100% ar 50% pardavimų įvykdymą. Visai kitokia situacija būtų, jei parduodama būtų viena prekė iš pvz. 1000 vienetų. Tada galima būtų prie parduodamų prekių pridėti vienetą, dalyti, ir tai iš esmės nepakeistų procentinės išraiškos ir netrukdytų teisingai suvokti rezultatus sudėtus į vieną lentelę.

Šiuo atveju duomenų prasmė neleidžia rezultatų pateikti kaip vieno sąrašo, kuriame jos būtų surūšiuotos pagal pardavimo reitingą. Duomenys turėtų būti pateikiami kaip du atskiri sąrašai - vienas prekių, kurios nebuvo parduotos, ir kitas – prekių, kurios jau buvo parduotos. Ir dar reikia sukurti prasmingą ir lengvai suvokiamą tokių duomenų pateiktį.

Panaudojus panašias į anksčiau aprašytas užklausas iš duomenų bazės išrenkami prekių peržiūros bei pardavimų duomenys, imant numatytuoju laiku visą duomenų kaupimo laikotarpį, nes ilgesnio laikotarpio rezultatai naudingesni. Gauti duomenys prasukus aplankytų prekių *foreach* cikle padalijami į du atskirus – iš vis neparduotų ir visų kitų parduotų prekių – masyvus, tuo pačiu suskaičiuojant jų reitingus (neparduotoms prekėms neskaičiuojamas) bei taip pat suskaičiuojamas parduotų prekių reitingų vidurkis, kuris vėliau panaudojamas reitingų imties standartinio nuokrypio reikšmei gauti.

Kadangi šios analizės tikslas yra pateikti prekes, kuriomis domimasi, bet neproporcingai mažai nuperkama arba visai nenuperkama, tai kaip pagrindinės problematiškos prekės patekiamos tos, kurios yra visai neperkamos, nuo didžiausio jų peržiūros skaičiaus mažėjimo kryptimi. Administratorius gali pasirinkti, kokio ilgio tokį sąrašą peržiūrėti. Ir taip pat atskiru sąrašu pateikiamos parduotos prekės jų reitingų mažėjimo kryptimi. Bet pateikti pilną tokių prekių sąrašą būtų neteisinga analizės tikslo prasme ir

duomenų suvokimo prasme. Nes didelė neparduotų prekių sąrašo dalis gali būti laikoma problematiška, tuo tarpu parduotų prekių sąraše problematiškumas turėtų būti nustatomas pasiremiant papildomais kriterijais. Kaip tokie kriterijai paimti reitingų imties vidurkis ir standartinis nuokrypis. Prekės, kurių reitingas nukrypęs į „blogąją“ pusę nuo vidurkio toliau kaip per standartinio nuokrypio reikšmę jau galėtų būti laikomos „problematiškomis“, bet šioje konkrečioje analizėje šalia neparduodamų prekių būtų teisingiau išvesti visas parduotas prekes, kurių reitingas nuo vidurkio nukrypęs į „blogąją“ pusę per dvigubą standartinį nuokrypį. Jų visada bus nedaug ir jų „problematiškumas“ derės su neparduotomis prekėmis.

Dar pastebėtina, kad PHP yra funkcijų skaičiuojančių imties vidurkius ar standartinius nuokrypius, tačiau jos yra papildomuose paketuose, kurių gali nebūti įdiegta serveryje, todėl patikimiau yra turimą masyvo reitingų stulpelį prasukti *for* cikle ir suskaičiuoti jo vidurkį ir standartinį nuokrypį.

Efektyviausiai parduodamos prekės

Šios analizės tikslas piešingas prieš tai buvusiai – siekiama nustatyti efektyviausiai parduodamas prekes, kad galima būtų priimti atitinkamus sprendimus dėl jų, pvz. užsakyti daugiau. Įgyvendinimo mechanizmas taip pat panašus: išsifiltruojami peržiūrėtų ir parduotų prekių duomenys, suskaičiuojamas reitingas, duomenys sudedami į masyvą, jis surūšiuojamas pasinaudojus *array_multisort()* funkcija ir prekių sąrašas išvedamas į ekraną. Tačiau skaitinės reitingo reikšmės čia išvedamos priešingu, „tikimybiniu“ santykiu, nes natūraliau suvokti didesnę reikšmę, kaip efektyvesnį pardavimą. Analizės kodas yra faile *efficient.inc.php* (pridedamas).

Susijusių prekių automatinio pateikimo mechanizmas

Vienas iš šio darbo uždavinių buvo sukurti statistinės analizės metodais paremtą automatinį mechanizmą, kuris pasiremdamas apdorotais statistiniais duomenimis automatiškai pateiks parduotuvės lankytojui pasiūlymus dėl kitų prekių, susijusių su jo peržiūrima preke. Kaip ir paprastoje parduotuvėje, taip ir internetinėje, prekes stengiamasi išdėstyti taip, kad perkant vienas, tuo pačiu būtų perkamos ir kitos, susijusios. www.ona.com parduotuvėje yra įdiegta galimybė administratoriui įvesti jo manymu susijusias prekes ir šios prekės vaizduojamos svetainėje, kaip galimai galinčios dominti pirkėją, peržiūrintį konkrečią prekę. Tačiau, kokios prekės pirkėjų akimis yra susijusios, ne visada galima atspėti ir numatyti. Ir čia gali padėti parduotuvės pirkėjų bei lankytojų veiksmų statistinė analizė.

Pirmiausia, galima būtų surinkti duomenis apie tai, kokias prekes kartu perka pirkėjai, kai jas perka ne po vieną. Jei kažkokios prekės perkamos kartu, galima teigti, kad jos

pirkėjo suvokime turi kažkokį ryšį. Koks tas ryšys, nustatyti nesiekiama. Pasitenkinama formaliu faktu, kad jos turi tokį pat ryšį kaip ir bet kurios kitos kartu perkamos prekės.

Panašų principą galima taikyti ir lankytojo navigacijos veiksmams. Jei asmuo pereina nuo vienos prekės prie kitos, reiškia jos turi ryšį. Jei per ilgesnį laiko tarpą susikaupia duomenų apie perėjimą nuo vienos prekės prie kitos, tai tas perėjimas turi didžiausią tikimybę būti pakartotas, jei jis vyksta dažniausiai. Kodėl taip vyksta, šiuo atveju taip pat neanalizuojama.

Sujungus bendrų pirkimų bei judėjimo tarp prekių duomenis galima būtų suskaičiuoti, kokias prekes pateikti lankytojui kaip susijusias su jo apžiūrima preke. Akivaizdu, kad kuo daugiau reikalingų duomenų sukaupta, tuo rezultatai gali būti tikslesni ir naudingesni.

Kol duomenų nebuvo daug, perėjimų tarp prekių skaičiavimo mechanizmas buvo realizuotas faile *markov.inc.php* (dabar ši dalis „užkomentuota“). Čia pritaikytas perėjimų skaičiavimo mechanizmas paremtas Markovo grandinių principu. Sukurta matrica, kurioje išfiltravus duomenų bazės *visitors* lentelę, įrašomi duomenys apie tai, kiek kartų atlikti atskiro lankytojo perėjimai tarp kiekvienos prekės. Matricoje, tiek stulpelių, tiek eilučių pavadinimai yra prekių numeriai, o jų susikirtimo langeliai rodo perėjimų skaičių iš eilutės prekės į stulpelio prekę. Gauta matrica nėra perėjimų tikimybių matrica, bet jos taikomoji prasmė šiuo atveju panaši: nustačius norimoje eilutėje didžiausias skaičių reikšmes, bus matyti, kur didžiausia tikimybė lankytojui judėti sekančiu žingsniu.

Šio skaičiavimo prasmė buvo daugiau demonstracinė, išvedanti į ekraną susijusius perėjimus, ir leidžianti stebėti, kaip veikia skaičiavimas.

Sukaupus duomenų bazėje jau daugiau duomenų apie lankytojų judėjimą, tie patys skaičiavimo principai pritaikyti įgyvendinant automatinį susijusių prekių pateikimo mechanizmą. Galima išskirti atskiras to mechanizmo dalis.

Duomenų bazė

Akivaizdu, kad kiekvieną kartą peržiūrint prekę skaičiuoti jos susijusių prekių matricą sudarytą iš kelių tūkstančių eilučių ir stulpelių, yra neracionalu serverio resursų atžvilgiu. Todėl naudinga būtų sukurti nuolat papildomą duomenų bazės lentelę su šiais duomenimis.

Tam tikslui sukurta lentelė *visitors_move* su keturiais stulpeliais: from_id (numeris prekės, iš kurios judama), to_id (numeris prekės, į kurią judama), count (perėjimų skaičius), sold (šių prekių pardavimo kartu skaičius). from_id ir to_id junginys sudaro pirminį raktą, kadangi tokia jungtis šioje lentelėje yra unikali.

Kadangi jau yra sukaupta duomenų, tai turimus duomenis naudinga būtų perkelti į šią lentelę. Faile *markov.inc.php (dalis MOVEMENT)* įrašytas kodas suskaičiuojantis perėjimus tarp prekių ir įrašantis rezultatus į naująją lentelę. Čia vėl panaudotos panašios į minėtas SELECT užklausos, o duomenys įrašomi pasinaudojant INSERT ... ON DUPLICATE KEY UPDATE... - pliusuoti perėjimų skaičių:

```
$sql = "INSERT INTO ona_visitors_move (from_id, to_id, count) VALUES ('0', '".$results[0]['product']."', 1) ON DUPLICATE KEY UPDATE count=count+1";
```

Taip pat įrašomi ir perėjimai į pirmąją prekę, įvedant, kad einama iš nulinės prekės į tą prekę (šie duomenys dabar nepanaudoti bet gali būti naudingi ateityje). Perėjimai skaičiuojami unikaliems lankytojams, sugrupavus juos pagal sesijos slapuką. Perėjimai iš tos pačios į tą pačią prekę lentelėje nesaugomi.

Kita kodo dalis faile *markov.inc.php (dalis SOLD PRODUCTS)* atlieka tos pačios lentelės stulpelio *sold* užpildymą. Principas panašus, skirtumas tik tas, kad čia nefiksuojama, iš kurios prekės, į kurią judama, svarbu tik, kad jos kartu parduotos, todėl užpildomos dvi eilutės – lyg judėjimo iš vienos prekės į kitą ir atgal.

Tokiu būdu lentelėje įrašoma, kiek kartų iš vienos prekės judėta į kitą ir kiek kartų jos kartu parduotos.

Siekiant turėti kuo daugiau duomenų, bendrų pardavimų duomenys paimti per visą parduotuvės veikimo laiką, o judėjimo duomenys – tik nuo 2011-01-08 (kai jie pradėti kaupti). Duomenų laikotarpių nesutapimas šiuo atveju nėra reikšmingas, tik reikia turėti galvoje, kad būtent pardavimų duomenys, jei tokių bus, ir labiausiai lems susijusių prekių vaizdavimą. Tai ir yra naudinga. Judėjimo duomenys atliks pagalbinį vaidmenį, kai nebus pardavimų duomenų, arba kai reikės įvertinti, kurią iš vienodai bendrai parduotų prekių vaizduoti.

Taip pat dar reikia paminėti, kad lentelėje sukuriamas „specialus“ prekių komplektas – judėjimas iš nulio į nulį, kuris skirtas saugoti santykio tarp prekių peržiūros ir pirkimo vidurkiui. Šis vidurkis kartu su standartinu nuokrypiu apskaičiuojamas faile *markov.inc.php* ir tuo pačiu įrašomas į lentelę. O adminisrtratoriui jis išvedamas lankytojų ataskaitų punkte „Statistika“. Kadangi vidurkio nebūtina nuolat atnaujinti, nes jis kinta lėtai, tai jo įrašymas nedarytas nuolatiniu, bet epizodiniu – kai administratorius ataskaitose jį pažiūrės (jis bus perskaičiuotas), tada jis ir bus įrašomas iš naujo.

Duomenų rinkimas

Vieną kartą įrašius senus duomenis, to daugiau daryti nebereikės, o lentelė bus nuolat pildoma kiekvieno kliento judesio metu. Tuo tikslu papildytas anksčiau minėtas failas

analysis.php, įrašantis duomenis į *visitors* lentelę. Čia sukurtas papildomas sesijos masyvo $_SESSION kintamasis *visitorsMove*, kuriame bus saugomos prieš tai buvusios prekės ir naujai aplankytos prekės numeriai. Kiekvieno šių įrašų pasikeitimo metu *vistors_move* lentelė bus papildoma nauju įrašu, panaudojant minėtą SQL užklausą INSERT ... ON DUPLICATE KEY UPDATE... .

Taip pat lentelė dinamiškai turės būti pildoma ir tuo metu, kai galutinai užbaigiamas vykdyti apmokėtas prekių užsakymas – tada pildomas *sold* stulpelis. Tuo tikslu modulyje, atsakingame už užsakymų tvarkymą, papildytas failas *actions.php* (papildymo kodas pridedamas). Kai užsakymas bus patvirtinamas kaip užbaigtas, šis kodas suveiks ir įrašys kiekvienos užsakytos prekės duomenis į lentelę *visitors_move* tuo pačiu principu kaip ir aprašytas duomenų perkėlimo metu.

Rezultatų pateikimas

Susijusių prekių pateikimas svetainėje organizuotas taip, kad administratorius gali įvesti iki 6 prekių, susijusių su redaguojama preke. Duomenys apie šias prekes įrašomi duomenų bazės prekių lentelėje viename laukelyje, įrašant kableliais atskirtus susijusių prekių numerius (įrašomi 8 skaičiai, bet naudojami tik 6, nes tiek prekių gražiau vaizduojama; pirmas ir paskutinis skaičius visada yra 0). Vaizduojant prekę naršyklėje, šis laukelis patikrinamas, ir tos prekės, kurių numeriai yra šiame laukelyje, atvaizduojamos kaip susijusios. Prieš tai dar patikrinama, ar kiti šių prekių parametrai leidžia jas vaizduoti. Tai atlieka prekių modulio failas *view_product.php*.

Kadangi administratorius visada gali turėti specialių tikslų vaizduoti konkrečias prekes kaip susijusias, tai diegiamas šis susijusių prekių vaizdavimo mechanizmas turėtų nekeisti duomenų bazės įrašų, o tik vaizduoti išskaičiuotas susijusias prekes tiek, kiek yra laisvų vietų tokioms prekėms pateikti, neliečiant administratoriaus įvestų prekių bei patikrinus, kad prekės nesikartotų.

Faile *view_product.php* neliečiant seno kodo įvestas papildomas kodas (pridedamas), kuris papildys susijusių prekių eilutės tuščias arba nuliais užpildytas dalis įrašais iš *visitors_move* lentelės, tolesnį šios eilutės apdirbimą paliekant senajai programai. 6 labiausiai susijusių prekių duomenys iš lentelės išrenkami tokia SQL užklausa:

```
$sql = "SELECT m.to_id, m.count+m.sold*(SELECT sold FROM ".PREFIX."_visitors_move WHERE from_id = 0 AND to_id = 0) AS countTotal FROM ".PREFIX."_visitors_move AS m, ".PREFIX."_shop_products AS p WHERE m.from_id = ".$d->id." AND m.to_id = p.id AND p.is_visible = '1' ORDER BY countTotal DESC LIMIT 6";
```

Iš jos matosi kam naudojama lentelėje įrašyta vidurkio reikšmė (saugoma kaip perėjimas iš nulio į nulį) – vidurkis dauginamas iš bendrai parduotų prekių skaičiaus ir prie jo

pridedamas judėjimų skaičius. Ir šis bendras skaičius, apjungiantis pardavimus ir judėjimus, yra kriterijus, pagal kurį atrenkamos labiausiai susijusios prekės. Taigi vienas bendras pardavimas yra prilyginamas vienam judesiui į šią prekę padaugintam iš vidutinio reikalingo peržiūrų kiekio, kad bet kuri prekė būtų nupirkta. Toks skaičiavimo būdas pasirinktas įvertinant jau minėtą aplinkybę, kad judėjimai tarp prekių yra papildomas kriterijus prie kartu parduotų prekių.

Susijusių prekių pateikties efektyvumo įvertinimas

Sukūrus veikiantį susijusių prekių pateikimo mechanizmą, būtų naudinga įvertinti jo efektyvumą. Svarbiausias efektyvumo įvertinimas būtų, jei pavyktų nustatyti, kiek padidėja kartu parduotų prekių kiekis, jei į jas patenkama per susijusių prekių pateiktį. Tačiau atsižvelgiant į parduotuvės specifiką, į tai, kad tokių kartu parduotų prekių būtų labai mažai per duomenų rinkimo laikotarpį, akivaizdu, kad rezultatai būtų labai nepatikimi dėl duomenų trūkumo. Bet galima panaudoti ir kitą būdą, kuriam duomenų jau pakaktų, – nustatyti, kiek dažniau lankytojai peržiūrinėja jiems pateikiamas susijusias prekes. Kadangi ir anksčiau svetainėje buvo pateikiamos susijusios prekės (bet įvedamos tik paties administratoriaus), tai paėmus santykį tarp susijusių prekių lankomumo dažnio, suskaičiuoto jau įdiegus automatinį pateikimą, su dažniu, kai susijusios prekės buvo įvedamos tik „rankomis“, gautume efektyvumo įvertį.

Techniškai tai realizuoti galima būtų imant atskiro lankytojo, nustatomo pagal *visitors* lentelės stulpelio ona_uid sesijos slapuką, judėjimą iš vienos prekės tiesiai į kitą. Kadangi svetainė yra taip sukomponuota, kad tiesiai, be tarpinių užklausų, iš vienos prekės į kitą galima pakliūti tik dviem būdais – per tame pačiame lange vaizduojamas susijusias prekes arba per judėjimo tos pačios kategorijos prekėse mygtuką, tai judėjimas iš vienos prekės tiesiai į kitą duos reikiamus rezultatus. Tačiau čia yra problema – svetainėje adresai formuojami taip, kad iš užklausos neįmanoma atskirti, ar eita į susijusią prekę, ar spaustas mygtukas „pereiti į sekančią tos pačios kategorijos prekę“; abi nuorodos suformuotos vienodu principu. Norint atriboti, galima būtų imituoti perėjimo į sekančią tos pačios kategorijos prekę skaičiavimą ir nustatyti, kokia buvo suskaičiuota ta sekanti prekė, bei lyginti su turimais duomenimis. Tačiau toks metodas galėtų būti gana netikslus ir todėl nepatikimas, nes nebūtų žinoma, kokios tuo metu prekės buvo apyvartoje, o jei buvo – ar jos buvo eksponuojamos, ir todėl šiuo metu suskaičiuotas vienos kategorijos prekių eiliškumas neatspindėtų tuo metu buvusio eiliškumo.

Bet iš kitos pusės, galima apsieiti ir be to. Pakankamai patikimus rezultatus įvertinti efektyvumui galima gauti ir atrenkant duomenis išmetus iš apskaitos visas užklausas į prekes,

priklausančias tai pačiai kategorijai. Tokiu būdu, aišku, būtų pašalintos ir užklausos į susijusias prekes, jei jos priklauso tai pačiai kategorijai. Tačiau skaičiuojant santykį tarp ankstesnio laikotarpio (nuo 2011-01-08, kai buvo pradėti kaupti duomenys, iki 2011-04-26, kai pradėjo veikti automatinis susijusių prekių pateikimas) ir naujo laiko periodo (nuo 2011-04-26 iki einamojo momento), galima teigti, kad jo pokytis dėl tokio pašalinimo nebūtų reikšmingas, ypač praėjus kuo ilgesniam laiko tarpui. Nes tiek ankstesniu, tiek naujesniu laikotarpiu tarp susijusių prekių santykinai vienodai dažnai gali pasitaikyti tų prekių, kurios priklauso tai pačiai kategorijai su pagrindine peržiūrima preke, todėl jų pašalinimas reikšmingai neįtakotų bendro santykio.

Susijusių prekių lankymo pokytis skaičiuojamas faile *markov.inc.php* ir rezultatas pateikiamas administratoriui ataskaitų dalyje „Statistika“. Pagrindinė sql užklausa duomenims atrinkti atrodo taip (panaudoti atskiri laikotarpiai užklausose atskiriems periodams):

```
$sql = "SELECT referer_product, product, ona_uid FROM (SELECT SUBSTRING_INDEX(SUBSTRING_INDEX(referer,'/',7),'/',-2) AS referer_product, SUBSTRING_INDEX(SUBSTRING_INDEX(request_uri,'/',5),'/',-2) AS product, ona_uid FROM ona_visitors WHERE time BETWEEN '2011-04-26 14:00:00' AND '".date("Y-m-d H:i:s")."' AND request_uri REGEXP '^/[a-z]{2}/products/[0-9]+/[0-9]+($|/.*)' AND referer REGEXP '^http://www.ona.com/[a-z]{2}/products/[0-9]+/[0-9]+($|/.*)' ORDER BY ona_uid, time) AS tt WHERE SUBSTRING_INDEX(referer_product,'/',1) <> SUBSTRING_INDEX(product,'/',1) GROUP BY referer_product, product, ona_uid ORDER BY ona_uid";
```

Kaip matyti iš užklausos duomenys apie tiesioginius perėjimus tarp prekių atrenkami iš *visitors* lentelės *referer* ir *request_uri* stulpelių, taikant šiek tiek skirtingus šablonus adresų išfiltravimui, nes jie saugomi lentelėje skirtinga forma. Be to išmetami ir to paties lankytojo tos pačios sesijos metu pakartotinai atlikti perėjimai tarp tų pačių prekių (panaudojant „GROUP BY“), kad pašalinti nereikšmingus „BACK“, „FORWARD“ ar „REFRESH“ judėjimus naršykle.

Gavus perėjimų į susijusias prekes skaičių už abu periodus ir suskaičiavus, kiek per atitinkamus laikotarpius buvo viso unikalių sesijų, atitinkamai padalijus perėjimų skaičių iš sesijų skaičiaus, gautume skaičių, rodantį, kiek vidutiniškai vienai unikaliai sesijai tenka perėjimų į susijusias prekes. Padalijus vėlesnio laikotarpio šį parametrą iš ankstesnio laikotarpio, gausime rezultatą, kuris rodys, kiek dažniau vėlesniu laikotarpiu vidutinis lankytojas iš prekės eina į vaizduojamą susijusią prekę. Ir, aišku, kad kuo ilgesnio laikotarpio duomenys imami, tuo labiau nereikšminga tampa atskirų galbūt labai išsisikiriančių lankytojų judėjimų įtaka.

Patikrinus duomenis 2011-05-21 dienai, judėjimas į susijusias prekes padidėjo šiek tiek daugiau nei 2 kartus: – 2,02 karto.

Išvados

- **Sukurta realiai veikianti sistema – pasiektas tikslas, įgyvendinti uždaviniai**. Siekiant šiame darbe užsibrėžto tikslo ir įgyvendinant iškeltus uždavinius buvo sukurta ir realiai veikiančioje elektroninėje parduotuvėje įdiegta lankytojų ir pirkėjų veiksmų statistinės analizės sistema (vaizdai priede B, veikianti sistema internete).
- **Sukurta sistema naudinga – gaunami realūs duomenys apie klientus leidžia priimti optimalesnius verslo sprendimus**. Sistemos dalis, analizuojanti lankytojų veiksmus ir pateikianti ataskaitas, yra akivaizdžiai matoma ir naudinga parduotuvės administracijai (kaip matyti iš UAB „Ona“ atsiliepimo, priedas C), siekiant sekti realią situaciją apie domėjimąsi prekėmis, klientų apsilankymus bei kitus duomenis ir, atsižvelgiant į tai, priimti verslo sprendimus. Per laiką, kai buvo pradėti kaupti duomenys, ir veikė keletas ataskaitų mechanizmų galima buvo stebėti, kad svetainė nėra aktyvi – joje naršo vidutiniškai apie 85 unikalių lankytojų per parą (laikotarpiu 2011-01-08 - 2011-04-25), per parą registruojasi 1-2 nauji klientai. Sukaupus duomenų jau matyti, kurios prekės susilaukia didžiausio susidomėjimo, nors nebūtinai jos aktyviai perkamos – kartais net visai neperkamos. Tokie rezultatai duoda peno administratoriams pamąstyti apie pardavimo strategijas. Kas ir daroma, ir galima teigti, kad duoda rezultatų – paskutiniu metu parduotuvės lankytojų aktyvumas didėja. Iš ataskaitų matyti, kad unikalių lankytojų nuo paskutinių įdiegtų pakeitimų (laikotarpiu 2011-04-26 – 2011-05-21) vidutiniškai padaugėjo apie 9,3 % - nuo 85,0 iki 92,9 per parą.
- **Padidėjusi susijusių prekių peržiūra**. Kita sistemos dalis – susijusių prekių skaičiavimas ir pateikimas lankytojui – nėra taip akivaizdžiai matoma, tačiau reikšminga, kaip metodas leidžiantis įtakoti lankytojų veiksmus ir dėl to naudinga, siekiant jiems parduoti vienas ar kitas prekes. Sukurtas ėjimo į susijusias prekes padažnėjimo įvertinimas rodo, kad įdiegtas susijusių prekių pateikimas veikia ir yra efektyvus. Lyginant beveik mėnesio, kai šis mechanizmas veikia, duomenis su ankstesnio laikotarpio (daugiau kaip trijų mėnesių) duomenimis, matyti, kad lankytojų judėjimas į susijusias prekes padidėjo daugiau kaip 2 kartus.
- **Sistema lengvai plečiama**. Sukurta parduotuvės klientų veiksmų analizės sistema yra nesunkiai plečiama: duomenų sukaupta ir toliau kaupiama daugiau negu kol kas panaudota, ataskaitų modulis leidžia lengvai prijungti naujas analizes ir jas pateikti

parduotuvės administracijai. Plečiamumas akivaizdžiai matomas ir šiame darbe, kai imantis naujo darbo etapo, nauja dalis lengvai integruojasi jau sukurtoje sistemoje.

- **Nustatyti tinkamiausi konkrečiai situacijai programavimo metodai**. Nors ir neaktyvi svetainė per keletą mėnesių sukaupė santykinai nemažą kiekį duomenų apie lankytojų veiksmus. O duomenų kiekio išsiplėtimas pareikalavo naudoti kitus programinius sprendimus nei buvo tinkami naudoti tada, kai duomenų buvo mažai, – efektyviau yra mažinti perduodamų tarp PHP variklio ir MySQL serverio duomenų kiekį, duomenų atrinkimą ir rūšiavimą perduoti MySQL serveriui, formuluojant kuo tikslesnes SQL užklausas ir gaunant pilnai ar beveik pilnai atrinktus reikalingus atsakymus.

Literatūros ir informacinių šaltinių sąrašas

1. Modules | Drupal.org. 2011-05-20. http://drupal.org/project/modules.
2. Michael Peacock. PHP 5 e-commerce Developement. Pact Publishing, Birmingham, UK: (14-15 p.). 2010.
3. Avinash Kaushik. Web Analytics – An Hour a Day. Wiley Publishing, Indianapolis, USA: (15-21 p.). 2007.
4. Justin Vincent – ezSQL. 2011-05-20. http://justinvincent.com/ezsql.
5. phemplate 2004. 2011-05-20. http://pukomuko.esu.lt/phemplate/slides/phemplate.xhtml.
6. Worx International Inc, PHPMailer, Tutorial. 2011-05-20. http://phpmailer.worxware.com/index.php?pg=tutorial.
7. Xdebug: Documentation. 2011-05-20. http://www.xdebug.org/docs/.
8. Jamie Huskisson. Magento 1.3: PHP Developer's Guide. Pact Publishing, Birmingham, UK. 2010.
9. Michael Peacock. PHP 5 e-commerce Developement. Pact Publishing, Birmingham, UK. 2010.
10. Toby Segaran. Programming Collective Inteligence. O'Reilly Media, Sebastopol, USA. 2007.
11. Justin Cutroni. Google Analytics. O'Reilly Media, Sebastopol, USA. 2010.
12. Martin Brampton. PHP 5 CMS Framework Developement. Pact Publishing, Birmingham, UK. 2008.
13. Dirk Merkel. Expert PHP 5 Tools. Pact Publishing, Birmingham, UK. 2010.
14. Todd Tomlinson. Beginnig Drupal 7. Apress, New York, USA. 2010.
15. Rasmus Lerdorf, Kevin Tatroe. Programming PHP. O'Reilly, Sebastopol, USA. 2002.
16. Peter B. MacIntyre. PHP: The Good Parts. O'Reilly, Sebastopol, USA. 2010.
17. Vikram Vaswani. PHP: A Beginner's Guide. McGraw-Hill, New York, USA. 2009.
18. Model View Controller (MVC) in PHP. 2011-05-20. http://php-html.net/tutorials/model-view-controller-in-php.
19. The Webalizer: FAQ. 2011-05-20. http://www.mrunix.net/webalizer/faq.html.
20. Eclipse – The Eclipse Foundation open source community website. 2011-05-20. http://www.eclipse.org.
21. Cookies & Google Analytics. 2011-05-20. http://code.google.com/intl/lt/apis/analytics/docs/concepts/gaConceptsCookies.html#HowGAUsesCookies.

Anotacija

Šiame darbe supažindinama su viena iš elektroninių parduotuvių tobulinimo krypčių – parduotuvės klientų veiksmų statistine analize, bei parodoma, kaip jau veikiančioje elektroninėje parduotuvėje galima įdiegti nesudėtingą tokios sistemos variantą. Nurodomos kryptys padėsiančios suprasti konkrečios elektroninės parduotuvės veikimą ir įdiegti joje klientų veiksmų analizės sistemą, pateikiamas įrankių aprašymas, apibūdinami duomenų apie klientus rinkimo metodai. Remiantis konkrečios elektroninės parduotuvės pavyzdžiu, parodoma, kokios statistinės analizės gali būti atliekamos ir kokią naudą jos galėtų duoti verslui, bei tyrimo eigoje sukuriama veikianti tokia analizės sistema apibūdinant tipiškas problemas, galinčias kilti panašiais atvejais. Taip pat aprašomas sukurtas mechanizmas, kuris remdamasis statistinės analizės metu gautais duomenimis, automatiškai generuoja susijusių prekių pasiūlymus pirkėjams.

Summary

The purpose of this study is to investigate the use of statistical analysis of e-shop traffic and to demonstrate how to integrate a fairly simple web data collection and analysis system into an already functioning e-shop, with a description of necessary programming tools and methods of collecting site traffic data. Using a real e-shop as a model, this study will show what kinds of statistical analysis can be introduced to an already functioning system, what benefits this can provide for the business, what problems typically arise during integration of statistical analysis systems, as well as how to solve said problems. The study will also investigate a system created to generate proposals for customers based on data analysis of customers' actions within the website.