VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
KOMPIUTERIJOS KATEDRA

Baigiamasis bakalauro darbas

Viešojo transporto keleivių srautų analizės sistema

Atliko:
Martynas Degutis parašas

Vadovas:
dr. Agnė Brilingaitė

Vilnius
2018

Turinys

Pratarmė

Šis baigiamasis darbas buvo atliktas su didžiuliu noru pritaikyti įgytas žinias bei kompetencijas realioje praktikoje, įgyvendinant dar nebandytus užsibrėžtus iššūkius. Įveikti šią asmeninę mokslinės srities užduotį padėjo ne tik asmeninės pastangos ar vidiniai norai.

Nuoširdžiai dėkoju už didžiulę pagalbą šio baigiamojo darbo vadovei, dr. Agnei Brilingaitei, kuri dalykiškai, tačiau teisingai bei taikliai sugeba paaiškinti daromas klaidas, suteikia naujų požiūrio krypčių bei priima studentą kaip lygiavertį asmenį. Abipusis ryšys gana sklandžiai leido vystyti mintis bei šio baigiamojo darbo eigą. Artimųjų bei mylimo žmogaus parama skatino vidinį atsakingumą bei nuoširdų rūpestį savo atliekamais veiksmais darbo eigoje. Jų supratingumas buvo emociškai pakilus bei jaukus.

Mano padėka skirta ir visų studijų metu dėsčiusiems dėstytojams. Kiekvieno iš jų pastangos pasiekti studentus per mokslą suteikė įvairių gabumų bei žinių, koks mokslinis pasaulis.

Sukurta sistema veikia Vilniaus Universiteto Matematikos ir Informatikos fakulteto debesų kompiuterijos suteikiamų resursų pagalba, nuotolinis pasiekiamas adresas: http://193.219.91.103:1995.

Santrauka

Atvirųjų duomenų paklausa leidžia stebėti bei analizuoti procesus, kuriuos kasdienybėje ne visada galima pamatyti ar atkreipti dėmesį. Šie duomenys suteikia galimybę bet kuriam asmeniui manipuliuoti besikeičiančia informacija, tai yra analizuoti, stebėti informacijos srautus bei pokyčius. Realius duomenys pateikiantys informacijos šaltiniai taip siūlo galimybę pažvelgti į informaciją iš asmeninės pusės.

Vilniaus miesto savivaldybės suteikiami šio miesto viešojo transporto duomenys parodo keleivius bei jų judėjimo kryptis. Iš šių duomenų galima kurti hipotezes apie žmonių judėjimo įpročius, galima stebėti, kaip pasiskirsto keleivių srautai tam tikru paros savaitės ar sezono metu. Taip pat keleivių srautų duomenys gali parodyti, kiek miesto gyventojams yra svarbus viešasis susisiekimas.

Šis darbas pateikia savitą keleivių srautų analizę, kuria remiantis, yra sukurta analizės internetinė sistema. Pastarąja gali naudotis bet kuris asmuo, kuris įgalinamas pateikti įvairią, su keleivių srautų judėjimu susijusią informaciją bei matyti rezultatą. Sistemos kūrimui pasitelkiamos įvairios informacinės technologijos bei įgyvendinimo būdai — nuo automatizuotų scenarijų rašymo įvairiomis programavimo kalbomis, iki dviejų skirtingų duomenų bazių valdymo ypatumų bei iššūkių juos atvaizduoti.

Summary

Web System for Public Transport Passengers' Flow Analysis

Open data, generally, is the idea of viewing data as a possibility to freely access it. This point of view focuses on letting other people get, edit and republish that publicly accessed data. These days the growth of open data is rapidly growing and more people have begun to value it as a confirmed information about any processed or information flow around the human population.

This paper presents the analysis part of specific open-access data which is provided by Vilnius city municipality. Main focus of this paper is to analyse the flow of city's public transport's passengers. In addition, it presents a mathod to automate the data obtainment, which is built on a server side, using scripts that are checking if open data is updated and converting it to match the database parameters. Main data flows are geographical and time space types which are used to describe where specifically the data should be depicted, and based on time relations.

Paper's analysis part is implemented programatically, as a one page application system. One depends on user's input parameters. The result of whole system is geographically shown passengers' flow on the virtual map. The flow is managed by the user itself, i.e. one is submitting parameters and the wanted result is shown graphically.

A system that ensures the liabilities to Vilnius city municipality to provide the newest data given. This practical form of manipulating the data flow lets users to see the real tracked information and analyse the movement of the majority of passengers. This system should be helpful as an analytical tool to influence future decisions, tied with public movement in the city. Desired city's open data of user movement could be easily applicable to the system aswell.

Įvadas

Dvidešimt pirmajame amžiuje sparčiai didėja miestų plėtra: į gyvenamuosius rajonus keliasi gyventi vis daugiau žmonių, plečiasi didžiųjų miestų ribos, gyvenimo kokybės pasiūla miestuose tampa vis labiau prieinamesnė. Drastiškai padidėjusios galimybės gyventi materialiai geresnį gyvenimą skatina žmones kurtis didžiuosiuose miestuose, idant greičiau bei patogiau pasiektų savo tikslų. Didžiosios dalies visuomenės materialinis gerbūvis leidžia šeimoms įsigyti vieną ar kelis automobilius, kuriais siekia pagerinti savo susisiekimą.

Viena pagrindinių gyvenimo mieste problemų — milžiniškos bei intensyvios eismo spūstys. Didžioji dalis gyventojų dėl asmeninio patogumo išvyksta į gatves nuosavais automobiliais, jų kelionių kryptys gana dažnai sutampa miestų ribose, todėl susidaro transporto eilės keliuose.

Viešojo transporto paslaugos — miestų sprendimas pagerinti žmonių susisiekimą. Jomis stengiamasi išspręsti eismo spūsčių problemą bei suteikti galimybę sparčiau keliauti gyventojams, negalintiems įsigyti asmeninės transporto priemonės. Dažna viešojo transporto praktika — autobusai, kurie talpina ne vieną dešimtį keleivių.

Tačiau taip pat kyla ir viešojo transporto problemų: nepakankamai gerai išvystyta kelių, pritaikytų viešajam transportui, infrastruktūra, eismo spūstyse atsiduria ir autobusai ar troleibusai. Didžiuliai keleivių srautai apsunkina sklandų transporto judėjimą. Dėl jų kyla ir asmeninių keleivių skundų patogumo ar asmeninio privatumo klausimais.

Gyventojai, norėdami pasiekti savo darbovietes ar kitus tikslus, privalo keliauti miesto gatvėmis, tačiau jomis keliauja ne vieni — kartu vyksta ir žymiai daugiau žmonių. Norint įgyvendinti sklandų keleivių srautų judėjimą, būtina kruopšti transporto bei žmonių judėjimo analizė, atsižvelgiant į gyvenamąją vietą, sukurtą miesto viešojo transporto infrastruktūrą bei alternatyvius keliavimo būdus — dviračių bei pėsčiųjų takų infrastruktūrą.

Šio darbo pagrindinis tikslas yra sukurti vieno puslapio aplikacinę platformą, kuri automatiškai atnaujintų Vilniaus miesto savivaldybės pateikiamus geografiniu bei laiko sąryšiu nurodytus duomenis, kuriuos šiuos atvaizduotų grafiniu pavidalu. Platforma leistų naudotojui aiškiau analizuoti miestų transporto infrastruktūros duomenis, jais manipuliuoti bei daryti išvadas.

Šiam tikslui pasiekti yra iškelti šie uždaviniai:

- Atvirųjų Vilniaus miesto savivaldybės pateikiamų duomenų analizė;
- Kintančių bei grafinių duomenų sąryšių modeliavimas;
- Nuo vartotojo įvesties priklausomas duomenų atvaizdavimas žemėlapyje;
- Užklausų kūrimas bei pritaikymas gaunamų įvesties parametrų atžvilgiu.

Įgyvendinus šiuos uždavinius bei išsikeltą tikslą, galima tiksliau bei kryptingiau vertinti viešojo transporto miestuose svarbą bei kokius šios paslaugos patobulinimus įmanoma pasiūlyti savivaldybei.

1. Susijusių darbų analizė

Šis darbas yra gana glaudžiai susijęs su viešuoju transportu, grafiniu duomenų atvaizdavimo praktikomis bei judėjimo trajektorijų analize. Šie trys punktai tarpusavyje turi sąryšių, kurie papildo vienas kitą bei detaliau gali apibūdinti.

Tiesioginį ryšį su šio darbo tema turi Vilniaus miesto savivaldybė. Pastarosios įkurta savivaldybės įmonė „Susisiekimo paslaugos“ yra atlikusi viešojo transporto keleivių srautų analizę[1] bei pateikusi ją grafiškai, su papildomais duomenimis. Šios įmonės anslitinėje platformoje yra pavaizduoti kintatys žemėlapiai, kuriuose atsinaujina besikeičianti informacija — punktais atskirti loginiai blokai, kurie pateikia žemėlapį ir ant jo pavaizduotus to bloko tematiką atitinkančius duomenis. Pavyzdžiui, pirmasis analizės blokas yra apie viešojo transporto stotelių paros apyvartą, kuriame pavaizduoti suminiai duomenys tam tikruose žemėlapio taškuose — stotelėse. Duomenys yra identifikuojami intuityviai, pagal spalvinę gamą bei figūros dydį (kuo didesnis apskritimas, tuo daugiau žmonių buvo toje stotelėje vidutiniškai per parą). Panašumai su šiuo darbu yra tokie, kad naudojami tie patys duomenys analizei atlikti. Esminis skirtumas — „Susisiekimo paslaugos“ sukurta platforma yra tik informacinio pobūdžio, nėra jokios vartotojui suteikiamos prieigos duomenų įvedimui bei užklausoms atlikti.

Šio darbo grafinės sąsajos pagrindas yra „Google Maps API“[2] programinė biblioteka, kuri suteikia galimybę asmeniškai susikurti žemėlapių sistemą bei pritaikyti juos asmeniniams tikslams pasiekti. „Google“ įmonė, kuri sukūrė šią intuityvią biblioteką, oficialioje įmonės žemėlapių platformoje pateikia nuasmenintus duomenis apie bet kokios rūšies transporto priemonių judėjimą gatvėmis. Platformos naudotojas gali stebėti realiu laiku besikeičiantį transporto judėjimą įvairiomis kryptimis. Įmonės šiems duomenims gauti pasitelkia vietos nustatymo parametrus[3], kurie įgalina pavienių asmenų vietos sekimą. Šie parametrai yra nustatomi pačių vartotojų asmeniniuose kompiuteriniuose ar mobiliuosiuose įrenginiuose. Šiame darbe duomenų rinkimas bei jų analizei parinktas būdas skiriasi nuo „Google“ naudojamos praktikos. Šiame darbe duomenys yra viešieji — plačiajai auditorijai internetu pasiekiamia duomenų rinkiniai, kurie yra pateikiami konkrečiuose geografiniuose taškuose, kurių pagalba netiesiogiai galima spręsti apie judėjimo trajektorijas.

Duomenų atvaizdavimas tiesiogiai priklauso nuo to, kokie duomenys yra gaunami. Geografinio pagrindo informacija ne visada yra tiksli — duomenys, priklausantys nuo GPS tinklo, gali būti su nedidele paklaida. Tuomet duomenų ribos yra nustatomos, atsižvelgiant į retus erdvėje esančius taškus, ir surenkami ta informcija, kurį tenkina ribos sąlygą. Kitas būdas — apjungiant didelius duomenų kiekius išskirti tam tikrą pasikartojančią duomenų struktūrą[4], kuri yra tankiausia, taip atsirūšiuojant nuo duomenų su paklaida. Šiame darbe duomenų vaizdavimo praktika yra paprastesnė. Duomenys tiesiogiai priklauso nuo vieno taško — stotelės — kuriame yra matuojami ir renkami keleivių judėjimo srautų duomenys. Turint konkretų tašką galima daryti tolesnes prielaidas bei analizuoti kiekybinius geografinių duomenų rezultatus.

2. Analizė

2.1. Vilniaus mieto savivaldybės atvirieji duomenys

Šiame darbe naudojami atvirieji Vilniaus miesto savivaldybės surinkti duomenys, kurie pagrindžia nuasmeninimo taisyklę ir viešai nepateikia duomenų, kurie padėtų identifikuoti bet kurį asmenį. Pagrindinis atvirųjų duomenų pateikimo tikslas — viešinti sukauptą informaciją tam, kad žmonės būtų informuoti apie vykstančius procesus, kurie nėra apibrėžti plačiajai visuomenei. Norima pateikti tokią informaciją, kuri galbūt nėra aktuali kiekvienam, tačiau būtų laisvai prieinama norinčiam domėtis kintančia aplinka.

Darbui atlikti aktualūs yra tik su viešuoju transportu susiję duomenys:

- Užfiksuojami kintantys duomenys
  - Keleivių srautai Vilniaus mieste 2016 metais
- Geografiniai duomenys
  - Vilniaus miesto bei seniūnijų administracinės ribos
  - Gatvių ir privažiavimų ašinės linijos
  - Viešojo transporto maršrutai
  - Viešojo transporto juostos
  - Viešojo transporto stotelių taškai

Pirmos rūšies — laiko sąryšiais susieti — duomenys yra naudojami norint pateikti viešojo transporto keleivių srautų apkrovimą, tai yra kokiais mastais yra keliaujama tam tikru maršrutu, kokie konkrečios stotelės keleivių skaičiai vyrauja tam tikru paros metu. 1 lentelėje nurodyti ne visi, bet esminiai stulpeliai. Pačiame faile yra 28-i stulpeliai, iš kurių 13-ka yra tušti, tai yra nepateikta jokia informacija. Stotelių duomenys yra sugrupuoti pagal mėnesius į atskirus failus.

Date	Time	Vehicle	Line	Direction	Stop number	...	Relat. fill. [ ]
2016-01-25	12:54:10	960	3	VAIDOTAI-STOTIS	76	...	11.63
2016-01-30	17:05:40	963	11	UZUPIS-ZVERYNAS	423	...	1.16
2016-01-18	22:26:14	958	74	STOTIS-PARKO G.	1207	...	6.98
2016-01-23	07:27:57	947	3	VAIDOTAI-STOTIS	5109	...	6.00

1 lentelė. Vilniaus sav. keleivių srauto duomenų fragmentas

Šioje lentelėje yra kelių rūšių kintamųjų: Date ir Time stulpeliai saugo datos tipo duomenis, Direction stulpelyje yra žodinės reikšmės (*string* formatas), *Relat. fill. [%]* saugo procentinės išraiškos skaičius dešimtainių formatu, o tokie stulpeliai, kaip Line ar Stop number — skaitinių reikšmių stulpeliai. Vilniaus savivaldybė *string* formato reikšmes saugo be lietuviškam raidynui priklausančių raidžių, todėl reikšmės nėra oficialios, jos pateiktos intuityviai suprantama išraiška, naudojantis tik lotyniškomis raidėmis.

Kito tipo duomenys — geografiniai. Juose yra saugoma koordinačių poros bei reliatyvi papildoma informacija, aprašanti tam tikrus, pavyzdžiui, stotelių ar kelių, objektus. Šio tipo duomenys Vilniaus miesto savivaldybė saugo *Keyhole Markup Language* formatu.1 išeities kodas atspindi failo fragmentą, kuriame pavaizduota vienos stotelės informacija.

Vilniaus savivaldybė šiuos duomenis saugo *string* formatu, išskyrus OBJECTID atributą, kuris yra skaitinės reikšmės ir saugo bendro naudojimo identifikacinį numerį. Pagrindinis atributas, kuris visoje sistemoje identifikuoja stotelę — KODAS atributas, kuris atitinka 1 lentelėje esantį

Stop number stulpelį. Per šį atributą yra galimybė susieti tiek geografinius, tiek laiko sąryšių duomenis.

Ne visi geografiniai duomenys yra aktualūs. Pavyzdžiui, STOTDATA atributas saugo stotelės įvedimo į bendrąją sistemą datą, o NUORODA — stotelės internetinį adresą. Tokie atributai yra tik papildomos informacijos šaltiniai, kurie neturi realios sąveikos su kitomis lentelėmis.

Šie geografiniai duomenys yra pateikiami dvejopai. Pagrindinis skirtumas — kaip pateikiamos koordinatės.1 išeities kode yra nustatyta vieno taško koordinačių pora <Point> elemento išraiška, o kituose failuose, kuriuose saugojama gatvių, administracinių ribų ir panašių linijinių geografinių duomenų, informacija yra pateikiama per <LineString> elementą. Šis saugo koordinačių porų sąrašą, kuris leidžia atvaizduoti taškus, sujungtus į vieną liniją.

1 išeities kodas. Vilniaus m. sav. stotelių duomenų fragmentas

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <kml xmlns="http://www.opengis.net/kml/2.2">
3 <Document><Folder><name>OGRGeoJSON</name>
4   <Placemark>
5         <ExtendedData><SchemaData schemaUrl="#OGRGeoJSON">
6                 <SimpleData name="OBJECTID">2286</SimpleData>
7                 <SimpleData name="PAVADIN">Grigaičių žiedas st.</
                      SimpleData>
8                 <SimpleData name="GATVE">Pergalės g.</SimpleData>
9                 <SimpleData name="KODAS">5163</SimpleData>
10                <SimpleData name="STOTDATA">2017-03-17T00:00:00.000Z</
                      SimpleData>
11                <SimpleData name="KRYPTIS">galinė st.</SimpleData>
12                <SimpleData name="NUORODA">http://stops.lt/5163</
                      SimpleData>
13        </SchemaData></ExtendedData>
14       <Point><coordinates>25.415546260482788,54.664296344384383</
             coordinates></Point>
15   </Placemark>
16 </Document></kml>
```

2.2. Laiko erdvės duomenų normalizavimas

Šiam darbui atlikti naudojami duomenys, kurie yra pertekliniai, tai yra — duomenys yra išskaidyti, dauguma reikšmių yra neužpildyta ir neturi jokios įtakos generuojant užklausas duomenų bazėje. Šie stulpeliai neatlieka savo funkcijos suteikti reikiamą informaciją.

2 lentelė atvaizduoja fragmentą pradinio atvirųjų duomenų failo, kuris gautas iš Vilniaus savivaldybės duomenų repozitorijos[5]. 2.1 poskyryje minėta, kad žodinės reikšmės duomenų faile yra išreikštos lotynišku raidynu. Šioje lentelėje galima įžvelgti net kelis trūkumus, kuriuos išsprendžia duomenų normalizavimas:

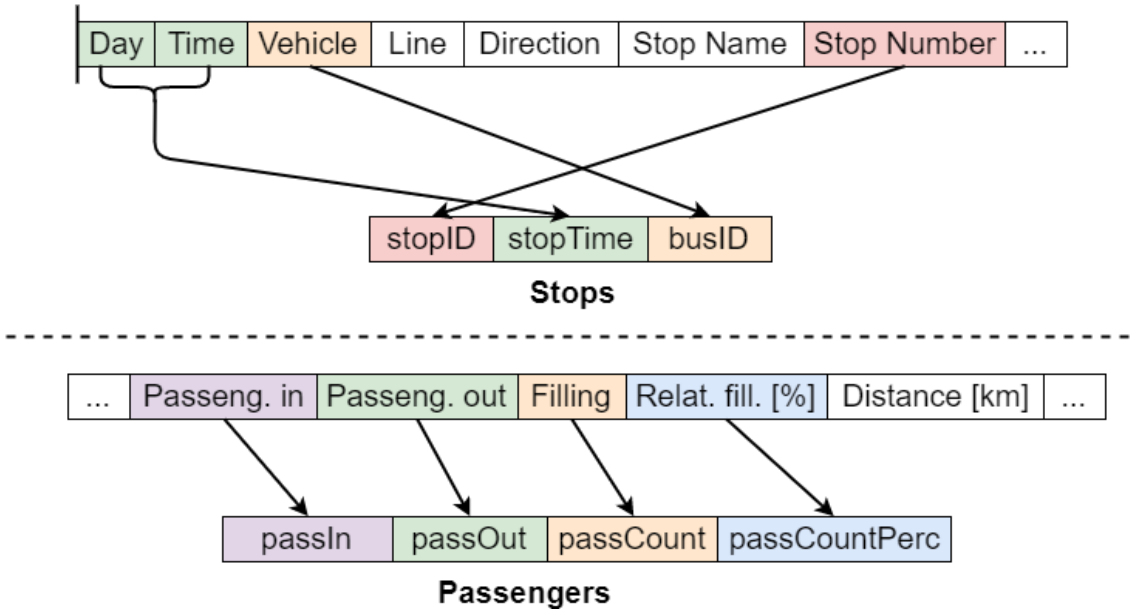
- Date bei Time stulpeliai — duomenų bazėje šių stulpelių reikšmės geriau apdorojamos apjungus į vieną stulpelį (*timestamp* tipas);
- Relat. fill. [%] stulpelis — nepakankamai aišku, kokia specifinė informacija yra saugojama, tad būtina pervadinti stulpelio pavadinimą;
- Pass. in [1] ir Pass. in [2] stulpeliai — šie nesaugo savyje jokių reikšmių, todėl jie sudaro papildomą apkrovą duomenų bazėje.

Day	Time	...	Stop name	...	Relat. fill. []	Pass. in [1]	...	Pass. in [2]	...
2016-01-05	00:00:40	...	VERKIU	...	0.00		...		...
2016-01-03	00:00:51	...	LATVIU	...	1.00		...		...
2016-01-11	19:38:48	...	RIESUTU	...	1.00		...		...

2 lentelė. Vilniaus sav. keleivių srauto duomenų fragmentas

Duomenys privalo būti normalizuojami tam, kad atliktų savo funkcinę priklausomybę, tai yra sugrupuoti pagal logines grupes, atskirti reikalingi tipai, kurie yra naudojami užklausų kūrimo etape. Denormalizavimo schema pavaizduota 1 pav. Jame atsispindi dviejų lentelių sūkurimas iš vieno, pradinio, failo.

Pirmoji lentelė **Stops** yra sukuriama iš keturių atributų, kurie tampa trimis stulpeliais. Stulpeliai Date ir Time tampa vienu, kadangi duomenų bazėje tipas, kuris apdoroja laiko formatą, yra *timestamp*. Šis tipas saugo kalendorinę bei laiko reikšmes kaip vieną konkrečią datą, pavyzdžiui, „*2016-01-05 14:51:47*“. Sukuriamam atributui stopTime nėra taikoma konkrečių ribų, duomenys egzistuoja nuo 2016-ųjų metų Sausio 1 dienos. Kiti atributai yra pervadinami ir jų reikšmės išlieka tokios pačios.



1 pav. Duomenų stulpelių formatavimas

Antrojoje lentelėje **Passengers** duomenys yra kopijuojami tik tie, kurie yra reikalingi, norint nusakyti keleivių kiekius. Atributai Passeng. in ir Passeng. out atitinkamai saugo įlipusių bei išlipusių vienoje stotelėje keleivių skaičių. Šios reikšmės yra skaitinės. Filling bei Relat. fill. [%] atributai saugo informaciją apie toje stotelėje bendrą esančių keleivių kiekį, atitinkamai skaitine ir procentine reikšmėmis.

1 pseudo kodas atspindi denormalizavimo eigą. Pradinis duomenų failas yra nuskaitomas, nustatomi reikiami atributų pavadinimai, kurie yra įrašomi į du naujus failus, atitinkamai reiškiantys **Stops** ir **Passengers** lenteles. Vėliau nuskaitytas failas yra iteruojamas eilutėmis ir į lentelių failus yra surašomos tik tos reikšmės, kurios atitinka nustatytus atributus.

1 algoritmas. Duomenų normalizavimo kodas

Data: Downloaded file of passengers' flows as X
Result: Stops as S and Passengers as P data tables' files
begin
    read X;
    define array $headerS$ for S; define array $headerP$ for P;
    write $headerS$ to S; write $headerP$ to P;
    let i be index of $headerS$; let y be index of $headerP$;
    for line $\in X$ **do**
        write $line[i]$ to S;
        write $line[y]$ to P;
    end
end

Rezultatas — denormalizuoti duomenys, kurie yra išskaidyti pagal funkcinę priklausomybę ir atrodo taip:

StopID	StopTime	BusID
2003	2016-01-28 00:00:00	942
2003	2016-01-19 00:00:02	965
311	2016-01-03 00:00:03	940
1917	2016-01-23 00:00:03	952
825	2016-01-20 00:00:05	965
1002	2016-01-31 00:00:05	932

3 lentelė. Denormalizuota **Stops** lentelė

PassIn	PassOut	PassCount	PassCountPerc
0	0	0	0.00
0	0	0	0.00
0	0	1	1.00
0	0	6	6.98
0	0	0	0.00
2	0	12	13.95

4 lentelė. Denormalizuota **Passengers** lentelė

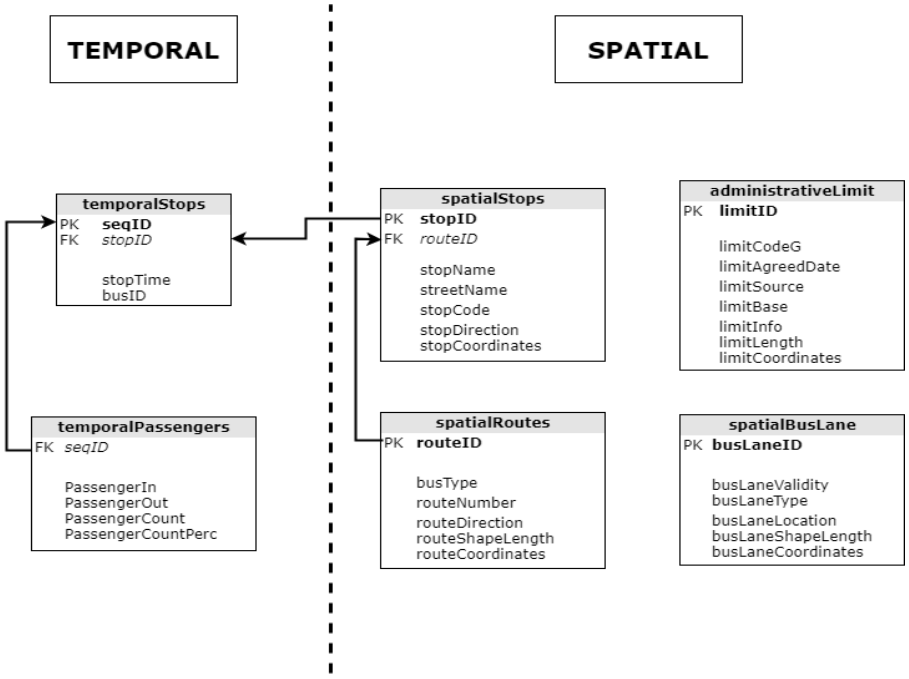
4 lentelėje matoma, kad StopTime atribute esantys laiko duomenys yra vientisi, kitos matomos reikšmės yra skaitinės, kai kurios kartojasi. Kadangi, pavyzdžiui, toje pačioje stotelėje (StopID) stoja ne vienas autobusas (BusID), šie duomenų srautai yra atskiriami pagal datą, kada autobusas sustojo toje stotelėje.

3 lentelėje didelių pokyčių nėra, tik yra atrinkti duomenys, kurie gali nusakyti keleivių srautų kaitą — kiek žmonių įlipo, kiek yra išlipusių ir koks bendras žmonių kiekis yra autobuse. Šie duomenys yra skaitinės reikšmės, išskyrus PassCountPerc, kuris saugo procentinę reikšmę — kokio apkrovimo tam tikrą akimirką yra autobusas.

Šiose lentelėse nėra nurodomas bendras identifikacinis numeris. Duomenys yra denormalizuoti taip, kad kiekviena eilutė abiejose lentelėse atitinka tą pačią eilutę iš pradinio duomenų failo. Identifikacinis numeris šiuo atveju yra generuojamas pačioje duomenų bazėje (2.3 poskyris).

2.3. Duomeų bazių modeliavimas

Duomenų srautą sudaro tiek geografiniai, tiek laiko erdvės duomenų tipai. Jie yra atskirti į dvi atskiras duomenų bazes — **Temporal** bei **Spatial** (2 pav.). Pirmojoje saugomi keleivių srautų kiekio bei erdvės duomenys, kitoje — pagrindiniai Vilniaus miesto savivaldybės geografiniai duomenys.



2 pav. Reliacinis modelis

Lentelės yra pavadintos pagal tai, kurioje duomenų bazėje jos egzistuoja, pavyzdžiui *spatialStops*. Ją galima identifikuoti pagal priešdėlinį žodį *spatial*. Tokia lentelė tokiu atveju yra **Spatial** duomenų bazėje.

spatialStops lentelėje saugomi Vilniaus miesto viešojo transporto stotelių geografiniai bei aprašantieji duomenys. Pirminis raktas **stopID** — savivaldybės nustatytas identifikacinis stotelės numeris. Išorinis raktas *routeID* — maršruto identifikacinis numeris. Taip galima atskirti, kurios stotelės kuriam maršrutui priklauso. Kiti atributai yra *string* formatu, jie apibūdina stotelę per pavadinimą, gatvę, kurioje yra stotelė, bei maršruto kryptį. Stotelės koordinatės saugomos **stopCoordinates** atribute. Tai yra — pirminiame faile 1 išeities kode nurodyto **<Point>** elemento koordinačių pora duomenų bazėje paverčiama į geometrinio tipo kodą, išreikštą skaičių bei raidžių seka (pavyzdžiui, *0101000020E61000003BA505E41C34394072E11532EA594B40* atspindi *(54.7024595839347;25.2035658372672)* porą).

Panašiu principu koordinatės yra saugojamos ir kitose **Spatial** duomenų bazės lentelėse. Skiriasi tai, jog koordinatės yra pateiktos kaip linijos koordinačių seka, bet išlieka koordinačių konvertavimas į geometrinio tipo duomenis koordinačių stulpeliuose.

spatialRoutes lentelėje yra pirminis raktas **routeID**, kuris yra sugeneruotas jau pradiniame faile. Jis yra *spatialStops* išorinis raktas. Kiti atributai nusako transporto tipą (**busType**), kuris juda

tam tikru maršrutu. Šią informaciją papildo **routeNumber** atributas. Pastarasis išskiria maršruto numerį, kurį priskiria Vilniaus miesto savivaldybė. Šis atributas nėra pirminis raktas šioje lentelėje todėl, kad savivaldybės suformuoti to paties numerio maršrutai yra pasikartojantys. Tai yra — maršrutai gali eiti kita kelio puse ar nukrypti į kitą gatvę, todėl skiriasi linijinė koordinačių porų seka. Tokie maršrutai yra saugojami atskirai, turintys tą pačią **routeNumber** atributo reikšmę.

spatialBusLane lentelė saugo autobusų juostų koordinates bei papildomą informaciją. Pirminis raktas **busLaneID** yra sugeneruotas savivaldybės. Visos kitos reikšmės, išskyrus **buslaneShapeLength** (saugo autobusų juostos linijos ilgį *double precision* formatu), yra *string* formatu. **busLaneValidity** apibrėžia galiojimo ribas, **busLaneType** nusako juostos tipą (pavyzdžiui, gali važiuoti ne tik autobusai, bet ir taksi autotransporto priemonės). Juostos koordinatės yra saugomos taip pat geometrinio tipo išraiška konvertuotos porų sekos.

Vienintelė lentelė, kuri nėra pavadinta su identifikaciniu priešdėliniu žodžiu — *administrativeLimit* lentelė. Šioji saugo tokius geografinius duomenis, kurie nėra susieti jokiais sąryšiais su kitomis lentelėmis. *administrativeLimit* lentelėje yra Vilniaus miesto savivaldybės administracinės miesto bei seniūnijų ribų koordinatės. Pirminis raktas **limitID** yra identifikacinis numeris, kuris yra sugeneruojamas duomenų bazėje.

Kitoje, **Temporal** duomenų bazėje yra dvi lentelės, kurios yra sugeneruojamos denormalizavimo būdu (2.2 poskyris). Duomenys šioje duomenų bazėje yra identifikuojami generuojant identifikacinį numerį **seqID**, kuris yra *temporalStops* lentelės pirminis, o *temporalPassengers* lentelės — išorinis raktas.

temporalStops lentelė yra sujungta su **Spatial** duomenų baze per *stopID* atributą, kuriame yra stotelių identifikaciniai numeriai skaitiniu formatu. per šį atributą apjungtos duomenų bazės įgalina galimybę laiko erdvės duomenis atvaizduoti žemėlapyje. Šiame darbe keleivių srautų informacija, priklausanti nuo laiko, atvaizduojama pagal stoteles, kadangi turimi duomenys leidžia stebėti konkrečios stotelės keleivių srautus.

Keleivių srautai ir tankiai yra aprašyti *temporalPassengers* lentelėje. Ji yra priklausoma nuo *temporalStops* lentelės per išorinį *seqID* raktą. Šie skaitiniai duomenys gali būti atvaizduojami žemėlapyje, nurodžius tam tikros stotelės geografinius duomenis. Duomenų tipai yra nurodyti 2.2 poskyryje.

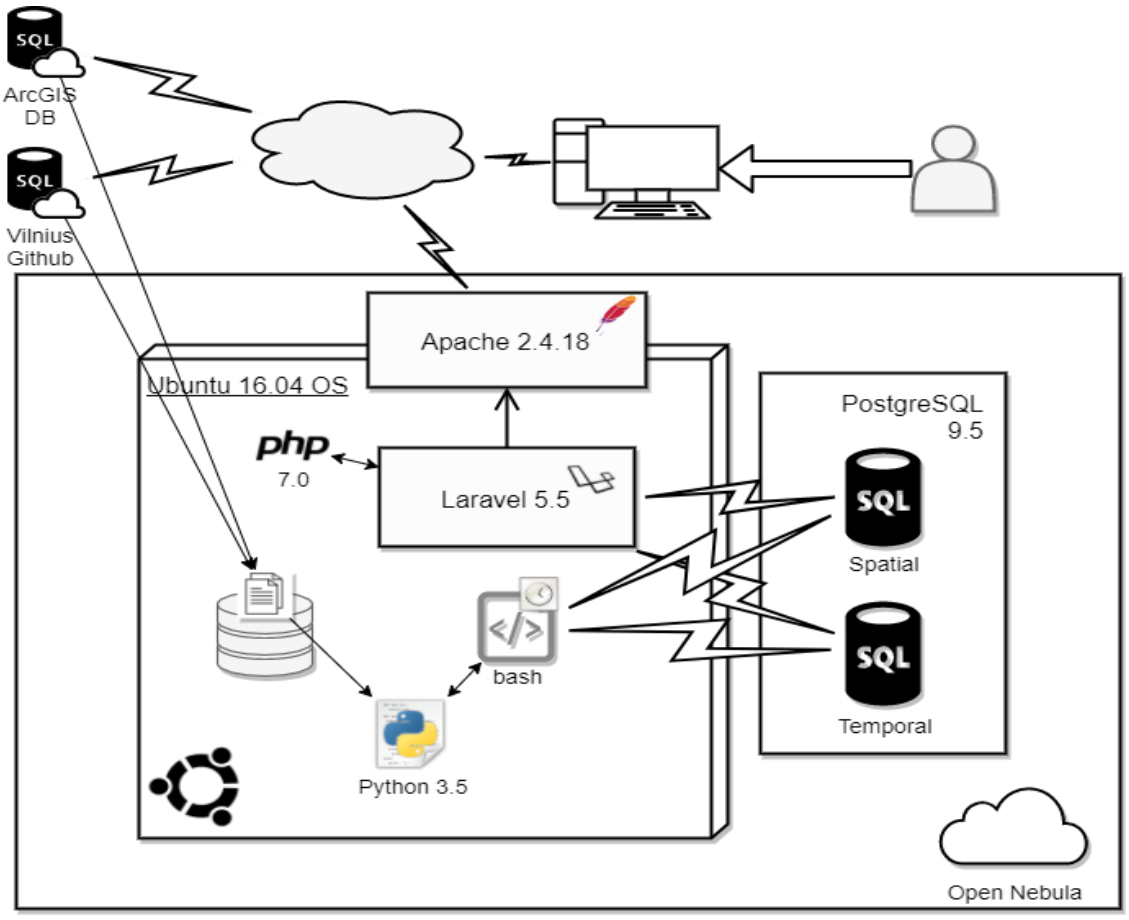
3. Sistemos architektūra

3.1. Naudojamos technologijos

Sistemos funkcinė architektūrai įgyvendinti pasirinkta standartinė „*LAPP*“ technologijų grupė. Į bazinį modelį įeina šios technologijos:

- Linux distribucinė operacinė sistema **Ubuntu**, *16.04* versija;
- *2.4.18* versijos HTTP serveris **Apache**;
- **PHP** programavimo kalba bei jos moduliai (versija — *7.0*);
- duomenų bazės valdymo sistema **PostgreSQL**, *9.5* versija.

Šios technologijos atsispindi 3 pav., kuriame yra pateikta visa sistemos naudojamų technologijų architektūrinė diagrama. Sistemai įgyvendinti bazinį technologijų modelį papildo kiti techniniai sprendimai.



3 pav. Architektūrinė diagrama

Pagrindinė sistemos dalis yra implementuota naudojantis Vilniaus Universiteto Matematikos ir Informatikos fakulteto resursais — debesų kompiuterijos paslauga **OpenNebula**. Joje yra sukurta virtuali mašina **Ubuntu** operacinės sistemos pagrindu. Virtualioje mašinoje įdiegtos bazinio „*LAPP*“ modelio technologijos — **PostgreSQL** duomenų bazių valdymo sistemoje yra implementuotos dvi duomenų bazės, bei HTTP serveris **Apache**, kuris padeda sistemai apdoroti HTTP servisus.

Sistemoje darbo tikslas — analitinė aplikacija — naudoja **Laravel** karkasą (*5.5* versija), kurios veikimas yra paremtas **PHP** programavimo kalba. Šis karkasas jungiasi prie HTTP serverio, kurio

pagalba aplikacija yra pasiekiama internetu bet kuriam vartotojui, kuris žino IP adresą bei tinklo prievadą bei turi kompiuterinę įrangą, galinčią prisijungti prie interneto. Duomenis aplikacijos įgyvendinimui **Laravel** gauna iš abiejų **PostgreSQL** sistemoje esančių duomenų bazių *Spatial* ir *Temporal*, prie kurių yra prisijungiama pačiame karkase.

Duomenų bazėms duomenis paskirsto ir įrašo **bash** komandinės eilutės apdorojimo kalba parašyti automatizuoti scenarijai, kurių eiga aprašyta kitame, 3.2 poskyryje. Pirminiai duomenys yra parsiunčiami iš Vilniaus miesto savivaldybės *Github* ir *ArcGIS* talpyklose pateiktų atvirųjų duomenų. Duomenys yra saugojami operacinės sistemos bendroje failų talpykloje. Šiuos failus apdoroja tiek **Python** programavimo kalba (*3.5* versija) parašytas normalizavimo scenarijus (2.2 poskyris), tiek **bash** sukurti scenarijai.

Naudojamų technologinių komponentų visuma sukuria unikalią sistemos architektūrą. Pastarąją galima konfigūruoti bei pritaikyti kitiems viešųjų geografinių bei laiko duomenų apdorojimui bei atvaizdavimui.

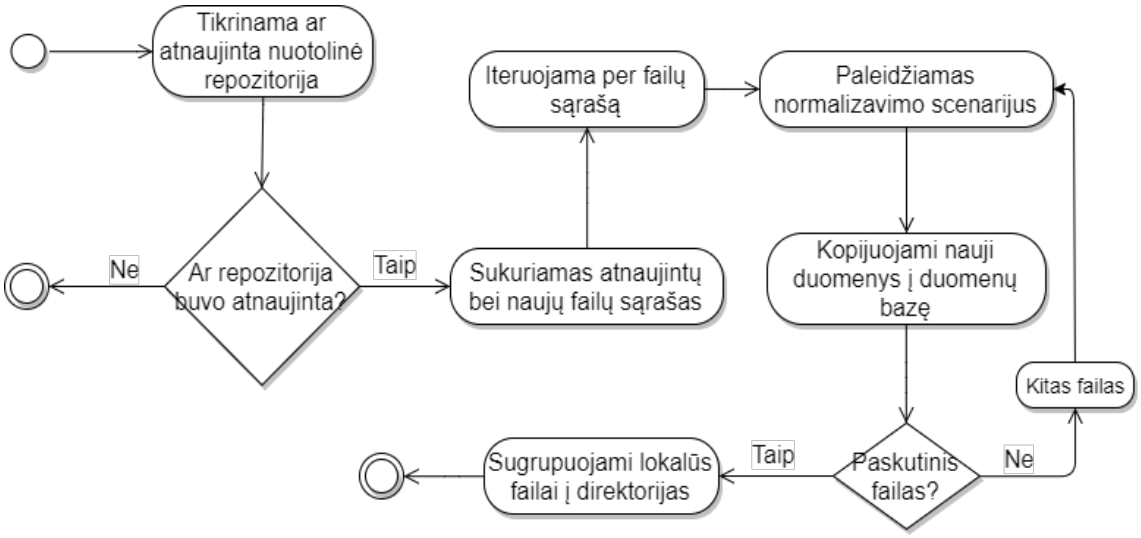
3.2. Automatizuotas duomenų atnaujinimas

Vilniaus miesto savivaldybė vienoje iš *Github* repozitorijų yra nurodžiusi tam tikrus įsipareigojimus, kuriuos privalo prisiimti asmuo, naudojantis savivaldybės pateiktus duomenis:

„Parsisiųsdami atvirus duomenis ir juos naudodami savo kuriamuose produktuose, programose, Jūs įsipareigojate kas kartą atnaujinti eismo tvarkaraščių ir kitą visuomeninio transporto aktualią informaciją, pagal tai, kaip tai yra pateikiama aukščiau esančioje nuorodoje.“

Laikantis šio įsipareigojimo, duomenis būtina reguliariai atnaujinti, kad pateikiama informacija išliktų pastoviai aktuali. Automatiškai renkami duomenys išpildo įsipareigojimus savivaldybei, kuri leidžia naudoti viešuosius duomenis asmeniniams tikslams pasiekti.

Pradinis duomenų atnaujinimo etapas — pirminiai duomenys yra tiesiogiai parsiunčiami į lokalią duomenų talpyklą. Geografiniai duomenys yra gaunami iš Vilniaus miesto savivaldybės *ArcGIS* duomenų bazės[6], kurioje talpinami įvairūs viešieji duomenys, susiję su transportu, teritorijomis ir kitais geografiniais duomenimis. Laiko erdvės duomenis Vilniaus miesto savivaldybė saugo vienoje iš *Github* duomenų versijavimo serviso repozitorijos.



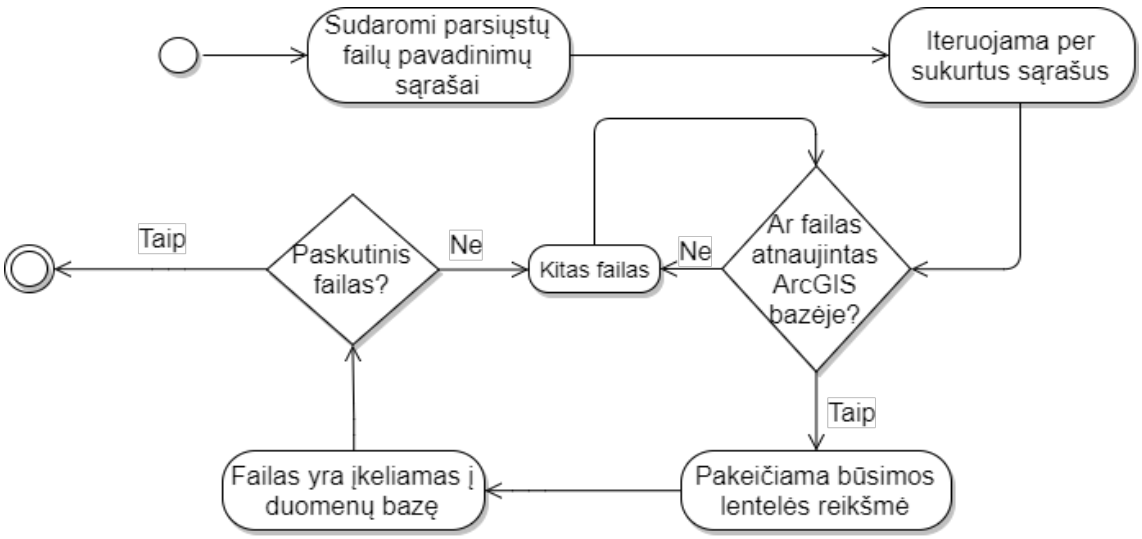
4 pav. Laiko erdvės duomenų atnaujinimo veiklos diagrama

Kiekvienai duomenų rūšiai apdoroti yra sukurti skirtingi **bash** komandinės eilutės apdorojimo kalba parašyti scenarijai. Kiekvienas scenarijus yra atsakingas už duomenų tikrinimą bei atnaujinimą atitinkamoje duomenų bazėje. Šie scenarijai yra aktyvuojami tam tikru nustatytu metu — kiekvieno einamojo mėnesio pirmąją dieną, 3-čią valandą nakties. Juos aktyvuoja **cron** — laiko nustatymais paremtas darbų tvarkymo įrankis. Operacinėje sistemoje šio įrankio pagalba nereikia aktvyuoti scenarijų veikimo rankiniu būdu, jie yra paleidžiami automatiškai.

Pirmąjį scenarijų **cron** įrankis paleidžia laiko erdvės duomenų apdorojimo veiksmams atlikti. Šio scenarijaus eiga — 4 pav. pavizduotoje veiklos diagramoje. Pirmajame žingsnyje yra tikrinama, ar Vilniaus miesto savivaldybės *Github* repozitorija buvo atnaujinta — jeigu nebuvo atnaujinta, tuomet duomenys taip pat nėra atnaujinami. Jeigu pirmojo žingsnio rezultatas yra teigiamas, kitame žingsnyje yra sukuriamas tekstinis dokumentas su naujų ar atnaujintų failų pavadinimų sąrašu, kuris yra iteruojamas.

Sąrašo iteravimo metu failas yra normalizuojamas (2.2 poskyris), naudojantis **Python** programavimo kalba parašytu scenarijumi. Kiekvienos iteracijos metu pastarasis yra paleidžiamas ir duomenys yra sutvarkomi. Kitame žingsnyje šie duomenys yra įkeliami į **Temporal** duomenų bazę, atitinkamai į *Stops* bei *Passengers* lenteles. Iteracijos pasibaigus, tikrinama, ar tekstiniame atnaujintų failų pavadinimų sąraše yra neapdorotų failų. Jeigu dar yra, prasideda kita iteracija ir duomenų apdorojimo bei įkėlimo ciklas kartojamas. Jeigu pasiekta sąrašo pabaiga, lokaliai sugeneruoti failai yra paskirstomi atitinkamai į *stops* bei *passengers* direktorijas, kuriose duomenys yra saugojami kaip atsarginės kopijos.

Lygiagrečiai kartu su pirmuoju scenarijumi, yra paleidžiamas yr kitas, geografiniems duomenims apdoroti parašytas scenarijus. Jo eiga — 5 pav. pavaizduotoje veiklos diagramoje. Pirmajame etape parsiųstų failų pavadinimai yra surašomi į tekstinius failus. viename faile saugomi lokalių failų pavadinimai, kitame — internetinės nuorodos, iš kurių yra parsiųsti atitinkami failai. Kitame žingsnyje vyksta šių sąrašų lygiagretus iteravimas. Pirmiausiai yra tikrinama, ar nuotolinėje Vilniaus miesto savivaldybės *ArcGIS* duomenų bazėje duomenų failas buvo atnaujintas. Jeigu ne, iteracija baigiama, pradedamas naujas kito failo tikrinimo ciklas.



5 pav. Geografinių duomenų atnaujinimo veiklos diagrama

Jeigu duomenys buvo atnaujinti, tuomet failas yra papildomas nauja informacija, pakeičiama lentelės **<Folder>** elemento reikšmė duomenų faile. Tuomet duomenys iš failo yra įkeliami į **Spatial** duomenų bazėje esančią atitinkamą duomenų lentelę, pagal nurodytą pavadinimą. Pasibaigus

iteracijai tikrinama, ar tai buvo paskutiniai failų sąrašų elementai. Po šio žingsnio, gavus teigiamą rezultatą, scenarijus baigia savo darbą arba, neigiamo rezultato pasekmėje, iteruoja sąrašą toliau. Duomenų atnaujinimo ciklas yra atnaujinamas su kitu elementu iš sąrašo.

Pasibaigus scenarijų darbui, duomenų bazėse esantys duomenys yra atnaujinami, papildomi pagal naujausią duomenų versiją, gautą iš Vilniaus miesto savivaldybės viešųjų duomenų resursų. Kas mėnesį tikrinami duomenys išpildo savivaldybės nustatytus įsipareigojimus bei palaiko kintamos informacijos aktualumą.

4. Sistemos įgyvendinimas

4.1. Užklausos

Sistemos kuriamos bei siunčiamos dviejų tipų užklausos: nuo vartotojo įvesties priklausomos bei statinės užklausos, kurios generuoja geografinių duomenų rezultatų atvaizdavimą. Internetinių svetainių kūrimo karkasas **Laravel** turi užklausų kūrimo įrankį *Query Builder* (liet. užklausų kūrėjas), kuris apsaugo duomenis nuo netinkamos įvesties į duomenų bazę. Kenkėjiška vartotojo įvestis gali sugriauti visą duomenų bazę ir karkasas pasirūpina, kad duomeny būtų saugūs.

4.1.1. Nuo įvesties priklausomos užklausos

Dinaminės užklausos yra generuojamos po vartotojo įvesties parametrų patvirtinimo. Vartotojas sistemos lange pasirenka norimus parametrus, juos patvirtina ir yra siunčiama užklausa į serverį. **Laravel** karkasas pasirūpina ryšiu su *Temporal* ir *Spatial* duomenų bazėmis ir įvykdo nurodytas duomenų bazių užklausas bei gauna rezultatą.

6 pav. pavaizduota pavyzdinė keleivių srauto reliacinės algebros formatu parašyta išskaidyta į kintamuosius užklausa. Šioje pavyzdinėje užklausoje yra apjungiamos trys lentelės — *temporalPassengers*, *temporalStops* bei *spatialStops*. Jos viena kitos duomenis pasiekia per atitinkamus identifikacinius numerius.

$$tables \leftarrow (temporalPassengers \times temporalStops \times spatialStops)$$
$$join \leftarrow \sigma_{temporalStops.id=spatialStops.id}(temp1)$$
$$date \leftarrow \sigma_{temporalStops.stoptime>'2016-06-04' \wedge temporalStops.stoptime<'2016-06-06'}(temp1)$$
$$limits \leftarrow \sigma_{temporalPassengers.passcount>0 \wedge temporalPassengers.passcount<=86}(temp1)$$
$$primary \leftarrow \sigma_{temporalPassengers.id=temporalStops.id}(temp1)$$
$$result \leftarrow \Pi_{AVG(temporalPassengers.passcount)}(primary \wedge limits \wedge date \wedge join \wedge tables)$$

6 pav. Keleivių srauto stotelėje vidurkio reliacinė algebra

Pirminiame žingsnyje yra surandami *temporalStops* ir *temporalPassengers* atitikmenys. Tuomet yra nustatomi tam tikri rėžiai ieškomam vieno autobuso talpinamam keleivių kiekiui apibrėžti. Tai yra privaloma užklausos grandis, kadangi žmonių autobuse negali būti mažiau negu neigiamas skaičius, ir negali būti daugiau negu autobusas gali maksimaliai talpinti. Vilniaus miesto savivaldybė yra nurodžiusi, kad nominali vidutinė autobuso keleivių talpa yra 86 keleiviai. Šis nustatytas keleivių kiekio intervalas apsaugo duomenis nuo išsikraipymo ir realybės neatitikimo.

To pasekoje yra apjungiamos *temporalStops* ir *spatialStops* lentelės ($join$ kintamasis). Geografinių stotelės duomenų lentelė yra reikalinga tam, būtų galima atvaizduoti gautą keleivių sumos stotelėje rezultatą prie atitinkamos stotelės koordinačių. Paskutiniame žingsnyje yra pasirenkamas *passcount* stulpelis — jis saugo autobuse esančių keleivių kiekį tam tikroje stotelėje.

6 pav. nurodytas $date$ kintamasis. Jo pasirinkimas yra neprivalomas, kadangi vartotojui leidžiama pasirinkti, ar pastarasis nori matyti gaunamą rezultatą tam tikrą dieną ar valandą, ar nori matyti bendrą visų duomenų vaizdą.

Kitoje, 7 pav. pavaizduotoje reliacinėje algebroje yra keleivių srauto sumos užklausa. Ši užklausa yra gana panaši į srauto vidurkio užklausą, skiriasi tik rezultatas — pateikiama tam tikroje stotelėje esančių keleivių kiekio autobuse bendra suma. $date$ kintamasis taip pat yra neprivalomas, jis padeda konkretizuoti duomenų srauto rezultatus.

$$tables \leftarrow (temporalPassengers \times temporalStops \times spatialStops)$$
$$join \leftarrow \sigma_{temporalStops.id=spatialStops.id}(temp1)$$
$$date \leftarrow \sigma_{temporalStops.stoptime>'2016-06-04' \wedge temporalStops.stoptime<'2016-06-06'}(temp1)$$
$$limits \leftarrow \sigma_{temporalPassengers.passcount>0 \wedge temporalPassengers.passcount<=86}(temp1)$$
$$primary \leftarrow \sigma_{temporalPassengers.id=temporalStops.id}(temp1)$$
$$result \leftarrow \Pi_{SUM(temporalPassengers.passcount)}(primary \wedge limits \wedge date \wedge join \wedge tables)$$

7 pav. Keleivių srauto stotelėse sumos reliacinė algebra

4.1.2. Statinės geografinių duomenų užklausos

Šios duomenų užklausos **Laravel** karkase yra implementuotos statiniu būdu. Geografiniai duomenys turi būti užkraunami kiekvieną kartą, kai vartotojas prisijungia prie sistemos. Tokiu atveju vartotojas gali matyti pradinę erdvę, kurioje gali atlikti analizę savo nuožiūra.

8 pav. pavaizduota pavyzdinė reliacine algebra išreikšta autobusų juostos koordinačių porų gavimas. Ši užklausa serverio pusėje yra iteruojama per visas *spatialBusLanes* lentelėje esančias autobusų juostas. $*id*$ atributas šioje algebroje yra iteruojamas, priskiriama vis nauja reikšmė iš visų esančių autobusų juostų, pagal $objectid$ atributą lentelėje.

$$select \leftarrow \sigma_{objectid=*id*}(spatialBusLanes)$$
$$lat - lng \leftarrow \Pi_{geometry.lat,geometry.lng}(select)$$

8 pav. Autobusų juostos (linijos) koordinačių porų reliacinė algebra

$geometry$ atributas yra geometrinio tipo atributas, kuris yra suformuojamas duomenų įdėjimo į *spatialBusLanes* lentelę metu (3.2 poskyris). Šį konvertavimą atlieka **PostgreSQL** domenų bazių valdymo sistemos plėtinys — *PostGIS*, kuris yra atsakingas už geografinių duomenų apdorojimą duomenų bazėje. Jo funkcionalumas leidžia iš geometrinio linijos tipo reikšmės išgauti koordinačių porų sekas (iš geometrinio taško tipo — vieną koordinačių porą).

Kiekvienos iteracijos metu koordinačių porų sąrašas yra saugomos bendroje autobusų juostų sekoje, kurios kiekvienos saugomas raktas — autobusų juostos identifikacinis numeris *objectid*. Pasibaigus visoms iteracijoms, koordinačių porų sąrašų seka yra grąžinama duomenų apdorojimui bei pateikimui žemėlapyje.

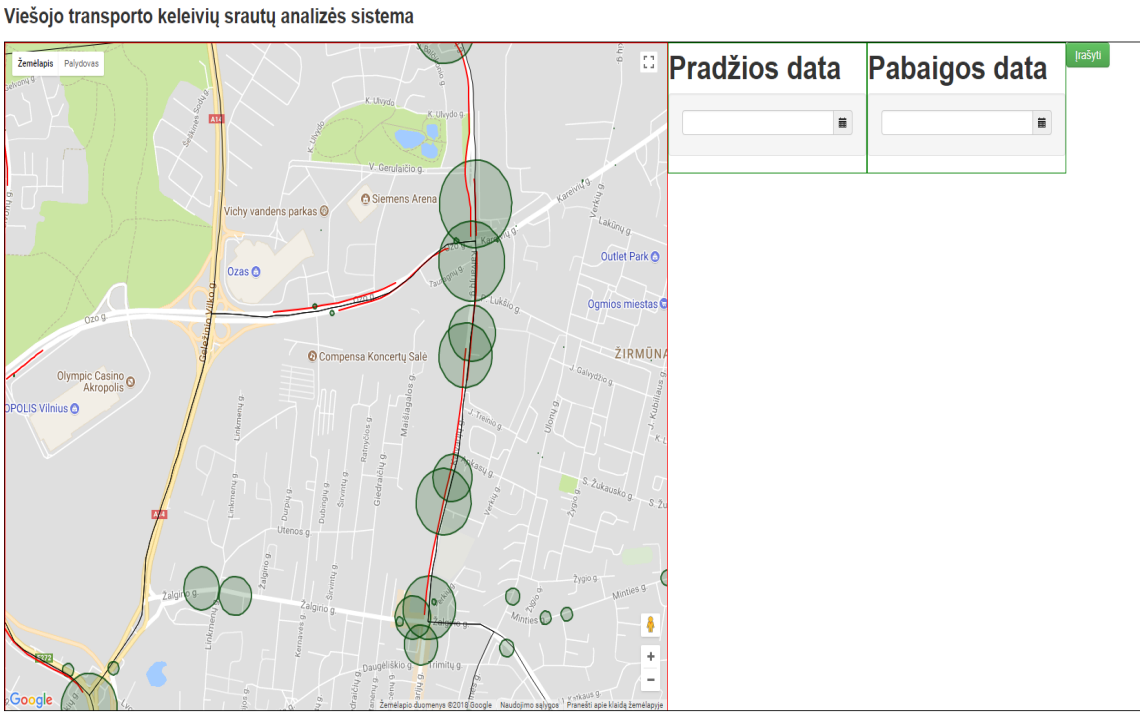
4.2. Vieno puslapio sistema

Sistema yra sukurta internetinių svetainių kūrimo karkaso **Laravel** pagrindu. Sistema nėra didelė, ji apdoroja vieną puslapį, kuriame yra: pagrindinis „**Google Maps API**“ bibliotekos pagalba implementuotas žemėlapis, bei įvesties laukai. Mobiliesiems įrenginiams ši sistema nėra pritaikyta todėl, kad didžiuliems duomenų srauto kiekiams apdoroti bei analizuoti patogiau yra darbalaukio versijoje.

9 pav. pavaizduota grafinė sistemos vartotojo sąsaja. Joje atsispindi kairėje pusėje esantis žemėlapis su esama grafine informacija. Jų reikšmės:

- Juodos spalvos linijos — Vilniaus miesto savivaldybės administracinės bei seniūnijų ribos;
- Raudonos spalvos linijos — autobusų juostų linijos;

- Žalios spalvos apvalios formos — užklausų rezultatai pagal rezultato dydį.



9 pav. Internetinės sistemos grafinis vaizdas

Šiame vaizdinyje yra rodoma atlikta suminė visų duomenų užklausa, kuri sumuoja visus keleivius tam tikrose stotelėse, esančius autobuso viduje. Toks duomenų atvaizdavimas įgalina lengviau pastebėti, kokie skirtumai yra tarp visų stotelių, leidžia palyginti stotelių apkrovą. Galima daryti prielaidas, kuriomis trajektorijomis vyksta intensyviausias eismo dalyvių judėjimas.

Figūros yra padaromos skaidresnės taip, kad matytųsi pirminis žemėlapio sluoksnis. Tokiu būdu galima matyti ir kitus, administracinius bei autobusų linijų ar kitus sluoksnius. Iš šios duomenų koreliacijos galima spręsti, kad autobusų juostos yra nutiestos tinkamuose keliuose, kadangi vyksta intensyvus keleivių srauto judėjimas. Duomenys nėra kintantys tol, kol nėra naujo vartotojo įvesties.

Išvados ir ateities planai

Visas šio darbo rezultatas nėra apibrėžiamas vieno uždavinio įgyvendinimu. Sistema, galinti analizuoti Vilniaus miesto savivaldybės ribose viešojo transporto keleivių srautų judėjimą, yra veikianti. Pirminiame etape išanalizuoti viešieji duomenys, įvertinti duomenų srautai bei pritaikyti duomenų apdorojimo žingsniai sukūrė tvirtus pagrindus vystytis toliau sukurtai sistemai.

Viešieji Vilniaus miesto savivaldybės duomenys nebuvo absoliučiai korektiški. Teko kai kurias reikšmes pritaikyti reikalingiems tipams, pavyzdžiui stotelių geografinių duomenų lentelėje identifikacinis numeris buvo *string* tipo, kurio negalima lyginti su skaitine reikšme kitoje, stotelės laiko erdvės duomenų lentelėje.

Pritaikytas automatinis duomenų atnaujinimas skirtingose duomenų bazėse, kuris išpildo Vilniaus miesto savivaldybės įpareigojimus atnaujinti esamą viešąją informaciją. Iš skirtingų duomenų bazių surenkami duomenų failai yra tuo pačiu saugojami operacinės sistemos vidinėje talpykloje, taip suteikiant galimybę grąžinti prarastus duomenis.

Duomenų bazių užklausos sukurtos taip, kad tenkintų kelis įvesties parametrus. Sistema yra įgalinta rinkti duomenis bei juos grafiškai atvaizduoti žemėlapyje.

Ateities perspektyvos šiai sistemai yra galimos, tačiau yra dalių, kurias galima patobulinti. Viena iš jų — maršrutų duomenų rinkimas ir atvaizdavimas, priklausomai nuo stotelių, kurios yra tame maršrute. Galima pritaikyti platesnį parametrų įvesties spektrą, tai yra atvaizduoti kelių rūšių duomenis (pavyzdžiui, stotelių srautų vidurkį bei keleivių kiekių sumą).

Taip pat naudinga būtų padidinti atliekamų operacijų spartą. Šiuo metu užklausos yra apdorojamos gana lėtai, užtrunka kelias minutes, priklausomai nuo norimo gauti rezultato. Tai būtų įmanoma padaryti optimizuojant užklausas, pritaikant taip, kad vartotų kuo mažiau sistemos resursų.

Vartotojo sąsaja įmanoma pritaikyti ir mobiliesiems įrenginiams taip, kad būtų patogus valdymas bei grafinis duomenų atvaizdavimas. Vartotojo langas galėtų atrodyt labiau priimtinas bei grafiškai stilistiškas, kadangi šia analitine sistema gali naudotis ne vienas žmogus nuotoliniu būdu.

Literatūra

[1] SĮ Susisiekimo paslaugos Vilniaus miesto savivaldybė. Vilniaus miesto viešojo transporto rodikliai, 2016. http://sisp.maps.arcgis.com/apps/MapJournal/index.html?appid=61fad018635947df9b80b6d45493ddfe#.

[2] Inc. Google. Google maps apis | google developers, 2018. https://developers.google.com/maps/.

[3] Tech Insider Tim Stenovec. Google has gotten incredibly good at predicting traffic — here's how, Dec. 18, 2015, 1:41 PM. http://www.businessinsider.com/how-google-maps-knows-about-traffic-2015-11.

[4] Z. Wang, T. Ye, M. Lu, X. Yuan, H. Qu, J. Yuan, and Q. Wu. Visual exploration of sparse traffic trajectory data. IEEE Transactions on Visualization and Computer Graphics, 20(12):1813--1822, Dec 2014.

[5] Vilniaus miesto savivaldybė. Keleivių srautai, 23 Kovo 2017. https://github.com/vilnius/keleiviu-srautai.

[6] SĮ „Vilniaus Planas". Vilniaus miesto savivaldybės vieši gis duomenys, 2016. http://gis-vplanas.opendata.arcgis.com/.