Latvijas Universitāte
Fizikas un matemātikas fakultāte
Datorikas nodaļa

Uzziņu dienesta 1188 datu bāzes pārveidošana un jauno iespēju realizācija

Bakalaura darbs

Autors
Anna Jevtušenko
DatZ020013

Vadītājs
M. Dat, SIA "Datorikas Institūts DIVI"
programmētāja- eksperte
Ija Šaporenkova

Rīga, 2006

Anotācija

Šajā darbā tiek apskatītas 1188 uzziņu dienesta sistēmas un to pakalpojumu raksturojums, detalizēti apskatot piedāvāto Interneta uzņēmumu katalogu un parādot tā iespējas. Aprakstīta uzņēmumu kataloga datu bāzes struktūra, kas, savukārt, satur informāciju par visiem Latvijā reģistrētiem uzņēmumiem un organizācijām, kā arī ģeogrāfiskajiem objektiem.

Ir apskatītas galvenās datu bāzes tabulas un datu bāzes automātiskās atjaunošanas procedūra, tās priekšrocības un trūkumi. Galvenā uzmanība tika pievērsta datu bāzes izmaiņu uzdevumiem un tās nepieciešamībai.

Tika veikta 1188 uzņēmumu kataloga datu bāzes pārveidošana, kas iekļauj izmaiņas datu bāzē un datu struktūrā un sastāv no diviem etapiem: datu atjaunošanas optimizācijas un jauno iespēju realizācijas. Pārveidošanas mērķis ir padarīt datu bāzes atjaunošanu ērtāku un ātrāku, papildināt katalogu Internetā ar iespējām apskatīt uzņēmumu atrašanās vietas un attēlot Rīgas pilsētas un piepilsētas maršrutus interaktīvajā kartē.

Ir dots uzlabota 1188 uzņēmumu kataloga Internetā salīdzinājums ar citiem līdzīgiem ārvalstu katalogiem.

Abstract

The work deals with the 1188 system and the services it provides. A detailed discuption is given of the directory of 1188 yellow pages on the Internet and the opportunities. The structure of the catalogue's database, which contains information of the enterprises and organizations registered in Latvia and geographical objects, is described.

The principal tables of the database, automatic database updating and its advantages and drawbacks are considered. The present work pays a great attention to purposes of database changes.

Transformation of the 1188 yellow pages's database has been accomplished. It includes database and date structure changes and consisted of two stages: optimisation of data updating and implementation of new features. The purposes of transformation were to make 1188 database updating faster and handy and to add the functions of positioning the enterprises and Riga city transport routes on the map.

A comparison of 1188 yellow pages with foreign analogoes catalogues is provided.

Аннотация

В данной работе рассматривается характеристика системы 1188 и её услуг. Подробно описаны возможности справочника предприятий 1188 в Интернете. Описана структура базы данных справочника предприятий, которая, в свою очередь, содержит информацию обо всех предприятиях и организациях, зарегистрированных в Латвии, а также географических объектах.

Рассмотрены главные таблицы базы данных и автоматическое обновление базы данных, его преимущества и недостатки. Наибольшее внимание уделено заданиям изменений базы данных и их необходимости.

Совершено преобразование базы данных справочника предприятий 1188, которое включает изменения в базе данных и структуре данных и состоит из двух этапов: оптимизация обновления данных и реализация новых возможностей. Цели преобразования: сделать обновление базы данных 1188 быстрее и удобнее, а также дополнить каталог в Интернете возможностями просмотра нахождения мест предприятий и Рижских городских и пригородных маршрутов на интерактивной карте.

Дано сравнение каталога 1188 в Интернете с иностранными аналогами.

Autoreferāts

Šis darbs ir veltīts 1188 uzziņu dienesta datu bāzes pārveidošanai un jauno iespēju pievienošanai Interneta uzņēmumu katalogam. Tika izstudēti un aprakstīti uzziņu dienesta piedāvātie pakalpojumi. Īpaša uzmanība tika pievērsta 1188 uzņēmumu katalogam Internetā un tā iespējām.

Tika veikta datu bāzes pārveidošana, kas ietver sevī divus etapus. Pirmais etaps iekļauj datu bāzes automātiskās atjaunošanas optimizāciju. Otrais - datu bāzes un datu struktūras izmaiņas, kas ir nepieciešamas, lai pievienotu jaunu funkcionalitāti. Jaunā funkcionalitāte ir iespēja meklēt uzņēmuma atrašanās vietu Latvijas interaktīvajā kartē pēc tā nosaukuma, kas praktiski nozīmē apskatīt uzņēmumu atrašanās vietu kartē un apskatīt Rīgas pilsētas un piepilsētas maršrutus šajā kartē.

Datu atjaunošanas gaitā tika izmainīta tabulu struktūra, kā arī uzrakstītas procedūras un trigeri, kuri veic visu nepieciešamu tabulu pārveidošanu. Izvēlēta metode ļauj glabāt izdarīto izmaiņu vēsturi, samazināt kļūdu rašanās risku, kā arī labot datu bāzes esošas kļūdas.

Lai būtu iespējams rādīt uzņēmumu atrašanas vietas interaktīvajā kartē ir nepieciešama tabulu pārveidošana, kas iekļauj sevī uzņēmumu un adrešu tabulu saiti. Šim mērķim tabulām tika pievienotas jaunas kolonas, kā arī uzrakstītas jaunās procedūras.

Maršrutu apskatīšanas realizācija tika veikta sadarbībā ar SIA Rīgas Satiksme. Šīs iespējas realizācija sastāv no vairākiem soļiem:

- tabulu struktūru veidošana, kas ļaus apskatīt pilsētas un piepilsētas autobusu, trolejbusu un tramvaju maršrutus, kā arī to pieturu un laiku sarakstus;
- datu eksportēšana no Rīgas Satiksmes datu bāzes (Microsoft Access datu bāzes) uz 1188 uzziņu dienesta datu bāzi;
- datu pārveidošana nepieciešamā formātā.

Uz doto brīdi projekts ir pabeigts, notestēts un tiek ekspluatēts.

Satura rādītājs

Ievads

Mūsdienās Internets kļūst visiem pieejams un bieži kalpo par visīsāko ceļu informācijas iegūšanai. Globālajā tīklā (Internetā) ir miljons dokumentu ar nestrukturizētu teksta informāciju. Lai atrastu vajadzīgo informāciju tīkla klientam, bieži vien, jāpārlasa simts Web lappušu. No 90. gadu sākuma strauji attīstās Interneta dienesti, kas palīdz atrast un apkopot vienā sarakstā vajadzīgo informāciju. Šajā darbā es plašāk pastāstīšu par vienu no tām - 1188 sistēmu, kā arī tās iespējām. Viens no galvenajiem sistēmas mērķiem ir glabāt un sniegt informāciju par visiem uzņēmumiem Latvijā.

Viens no svarīgākajiem Interneta kataloga uzdevumiem ir ātra un ērta informācijas meklēšana, kas ir atkarīga no datu bāzes struktūras labas organizācijas. Tādēļ, kā sava pētījuma mērķi, es izvēlējos datu bāzes izpēti un tās pārveidošanu.

Savlaicīga datu atjaunošana ir ļoti svarīga jebkurā informatīvā sistēmā. Lielās datubāzēs atjaunošana var izraisīt lielas grūtības, jo prasa daudz laika un cilvēka pūļu. Tika izdarīti soļi, kas atvieglo šo procesu 1188 datu bāzē un padara to daudz ātrāku. Datu bāzes atjaunošana nav vienīgais, kas ietekmē cilvēka interesi konkrētam uzņēmumu katalogam Internetā, piemēram, ļoti svarīgi ir piedāvāt lietotājiem daudz dažādu iespēju. Arvien vairāk cilvēku sāka lietot gan uzņēmumu katalogu, gan adrešu meklēšanu kartē, kā rezultātā kļuva aktuāls jautājums par jauno iespēju pievienošanu: radīt katalogā atrasto uzņēmuma atrašanas vietu Latvijas kartē, kā arī meklēt kartes fragmentu pēc uzņēmumu nosaukumiem. Šo iespēju nevar pievienot bez datu bāzes struktūras izmaiņām, jo viena tāda pieprasījuma izpildīšanai nepieciešams ļoti daudz laika. Tātad otrais darba uzdevums ir saistīt uzņēmuma adreses ar konkrētiem punktiem kartē.

Interaktīvās kartes ir diezgan jauns un strauji attīstošs virziens Latvijas Internetā, kas ļauj apceļot Latviju, nepieceļoties no sava krēsla. 2001. gadā "Karšu izdevniecība Jāņa sēta" saviem klientiem piedāvāja pirmo interaktīvo karšu sistēmu Latvijā. Tagad jau apmēram 10 Latvijas portālos var apskatīt valsts lielākās pilsētas kartē, kā arī meklēt dažādus objektus tajās. 1188 Uzziņu dienests nolēma paplašināt iespēju sarakstu savās interaktīvajās kartēs un sadarbībā ar SIA Rīgas Satiksmi izstrādāt sistēmu, kas ļaus lietotājiem apskatīt autobusu, trolejbusu un tramvaju maršrutus Latvijas kartēs, kā arī to pieturu un atiešanas laiku sarakstus. Līdz ar to pēdējais uzdevums ir izveidot datu bāzes struktūru, kas nodrošina šīs iespējas realizāciju.

Darba pirmajā daļā aprakstīti dažādi 1188 piedāvātie pakalpojumi: 1188 uzziņu dienests-uzziņu saņemšana pa tālruni, 1188 tālruņu katalogi, 1188 uzņēmumu katalogs Internetā un citi. Īpaša uzmanība tika pievērsta uzņēmumu katalogam Internetā, tā vēsturei, sadaļām un iespēju apskatīšanai. Tiek paskaidrota tā augošā popularitāte un apskatītas galvenās kataloga priekšrocības un trūkumi.

Darba otrā daļa dod vispārējo priekšstatu par 1188 datu bāzes struktūru. Šī daļa paskaidro kāpēc par datu bāzes serveri tiek izmantots Sybase Adaptive Server Enterprise un apraksta tā nozīmīgākās īpašības, kā arī satur 1188 datu bāzes tabulu un atjaunošanas procesa aprakstu.

Uzņēmumu kataloga uzdevumi ir dod aktuālo informāciju par uzņēmumiem, tas nozīme, ka ļoti svarīgi ir savlaicīgi atjaunot datu bāzi, kā arī piedāvāt daudz dažādu iespēju. Sakarā ar daudzām esošām 1188 datu bāzes atjaunošanas problēmām un trūkumiem bija pieņemts lēmums par datu bāzes automātiskas atjaunošanas optimizāciju un datu bāzes pārveidošanu, kas ļaus pievienot jaunās iespējas 1188 uzņēmumu katalogam Internetā.

Trešajā daļā sniegts pašreizējo problēmu un trūkumu raksturojums, kā arī ir dots plānoto izmaiņu virspusējs apskats.

Ceturtā darba daļa apraksta tieši datu bāzes pārveidošanu, kas sastāv no datu atjaunošanas optimizācijas, izmaiņām adrešu un uzņēmumu tabulās un izmaiņām, kas ir saistītas ar maršrutiem, un veiktos soļus.

Piektajā daļā ir dots uzlabota 1188 uzņēmumu kataloga Internetā salīdzinājums ar citām līdzīgām Interneta sistēmām.

Darba mērķi – padarīt 1188 datu bāzes atjaunošanu ērtāku un ātrāku un papildināt katalogu Internetā ar iespējām apskatīt uzņēmumu atrašanās vietas un Rīgas pilsētas un piepilsētas maršrutus interaktīvajā kartē - ir sasniegti. Tagad datu bāzes atjaunošana notiek vairākas reizes ātrāk un 1188 uzņēmumu katalogs Internetā strādā ar uzlabotu datu bāzi, kas ļauj izmantot jaunās iespējas.

1. 1188 piedāvātie pakalpojumi

Mūsu informācijas sabiedrībā cilvēki, lai apmierinātu konkrētās informatīvās vajadzības, aizvien biežāk izvēlas kvalitatīvus un ērtus informācijas avotus – tādus, kuros liels informācijas apjoms ir vienkopus – uzziņu dienestus, Interneta datu bāzes un tālruņu katalogus.

1996. gada rudenī, kad darbu sāka Uzziņu dienests 118, tajā strādāja 20 operatori, kas dienā apkalpoja apmēram 200 zvanu. 2005.gada 15.decembrī saskaņā ar dažādu Eiropas sakaru organizāciju rekomendācijām numurs 118 tika mainīts uz četrciparu numuru 1188. Šobrīd Kontaktu centrs 1188 ir lielākais informatīvais un operatoru centru pakalpojumu sniedzējs Baltijas valstīs. Tas ir uzziņu dienests, kura vissvarīgākais mērķis ir sniegt informāciju par visu Latviju jebkurā diennakts laikā. Uzziņas var dabūt pēc dažādiem parametriem- specializācijas, adreses, nosaukuma, tālruņa numura (juridiskām personām), darba laika [1].

1.1. 1188 uzziņu dienests

1188 piedāvā ļoti dažādus informatīvus pakalpojumus, visvecākais no kuriem ir uzziņu saņemšana pa tālruni. Dienā Uzziņu dienests apkalpo 52 000 zvanu, bet 2005. gada pirmajos desmit mēnešos apkalpoto zvanu skaits ir sasniedzis 11,9 miljonus, un tas ir par 17% vairāk nekā tādā pašā laika periodā 2004. gadā [1].

Piezvanot pa tālruni 1188 var uzzināt sekojošu informāciju:

Tālruņu uzziņas

- Firmu, uzņēmumu un iestāžu tālruņu numuri (adreses un darbalaiks)
- Privātpersonu tālruņu numuri
- Kodi Latvijā un ārzemēs
- Zvanīšanas kārtība Latvijā un uz ārzemēm
- Tiešais savienojums

Sabiedriskā transporta uzziņas

- Vienīgais oficiālais autobusu kustības saraksts
- Pasažieru vilcienu kustības saraksts
- Lidmašīnu kustības saraksts
- Baltijas jūras prāmju kustības saraksts
- Taksometra izsaukšana

Ceļojumu uzziņas

- Tūrisma iespējas
- Viesnīcas, viesu nami, nakšņošana
- Operatora palīdzība orientēties jebkurā Latvijas pilsētā
- Attālumu starp lielākajām Eiropas un Latvijas pilsētām.
- Informāciju par valstīm – galvaspilsēta, platība, naudas vienība, vīzu (bezvīzu) režīms, iedzīvotāju skaits u.c.

Izklaides uzziņas

- Kino, teātru, koncertu, operas un klubu repertuārs
- Filmu anotācijas
- Sporta pasākumi, vieta, laiks, rezultāti
- Informācija par dažādiem pasākumiem, izstādēm, konferencēm visā Latvijā
- TV un radio programma
- Nedēļas un dienas horoskopi
- Vārdadienas
- Svētku dienas

Finanšu uzziņas

- Latvijas banku kodi
- Banku valūtas kursi
- Bankas automātu adreses Latvijā un izmantojamās kartes
- Rīgas Fondu biržas kārtējās sesijas rezultāti

Vispārējas uzziņas

- Latvijas pasta indeksi un pasta pakalpojumu izmaksas
- Informācija par dažādiem mācību kursiem
- Laika prognoze – šodien, rīt, parīt
- Mērvienības
- Jebkura nestandarta informācija

Uzziņas var saņemt latviešu, krievu, angļu, vācu valodās [2].

Visjaunākā no iespējām, kas ir pieejama tiem, kas zvana uz 1188 uzziņu dienestu ir telemārketings, kas aptver:

- telefonaptaujas
- pārdošana pa telefonu
- klientu pasūtījumu pieņemšana
- klienta informācijas tālruņa apkalpošana
- pakalpojuma zvans pasūtītāja klientiem
- ielūguma serviss
- klientu datu bāzes serviss
- klientu informatīvo materiālu izplatīšana
- klientu sarakstu sagatavošana [3]

1.2. 1188 tālruņu katalogi

Kopš 1997.gada 1188 sadarbībā ar Lattelekom SIA sagatavo un publicē privātpersonu un uzņēmumu tālruņu katalogus "Rīga, Rīgas rajons, Jūrmala", "Vidzeme", "Kurzeme", "Zemgale",

"Latgale", kā arī uzņēmumu tālruņu katalogus "Latvija+" [4].

1188 tālruņu katalogi aptver konkrētu ģeogrāfisku teritoriju – Vidzemi, Kurzemi, Zemgali, Latgali, Rīgu, Rīgas rajonu un Jūrmalu, un satur pilnīgu informāciju par uzņēmumu un privātpersonu tālruņu numuriem, adresēm, kā arī klasificētu biznesa informāciju par uzņēmumiem noteiktajā ģeogrāfiskajā teritorijā. Katalogi tiek izdoti reizi gadā un tiek izplatīti bez maksas.

Izmantojot tālruņu katalogu Latvija+ un tā elektronisko versiju E-REĢISTRS, var lietot informāciju par visiem ekonomiski aktīvajiem (vairāk nekā 56 000) uzņēmumiem, organizācijām un valsts iestādēm Latvijā [5].

E-reģistrs ir elektroniska datubāze CD ROM formātā.

E-reģistrs dod iespēju:

- meklēt informāciju pēc uzņēmuma nosaukuma, tālruņa numura un teksta fragmenta;
- atlasīt uzņēmumus pēc to darbības veida un atrašanās vietas (pilsētas, rajona, ielas);
- automātiski nosūtīt informāciju pa e-pastu gan vienam, gan vairākiem uzņēmumiem;
- drukāt kontaktinformāciju sarakstu veidā;
- drukāt adrešu uzlīmes izvēlētiem uzņēmumiem.

Nepārtrauktā komunikāciju attīstība un straujais dzīves temps ir pamudinājis uzņēmumu pilnveidot un izvērst savu darbību arī citos informatīvās industrijas virzienos.

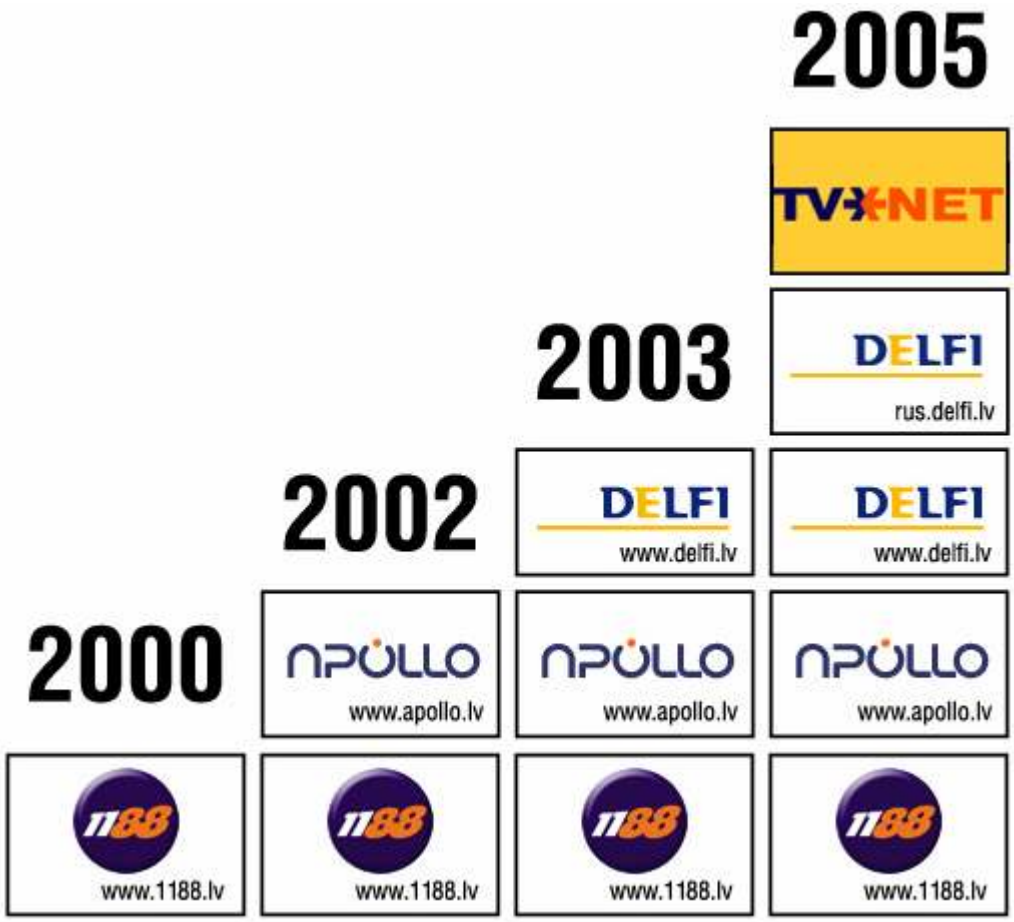
No 2000.gada 1188 piedāvā arī savu lappusi Internetā. www.1188.lv (līdz 2005.gadam www.118.lv) portālā ir pieejami: čats, e-pasts, sms sūtīšana un citas funkcijas. Tomēr visgalvenākā iespēja ir 1188 uzņēmumu katalogs Internetā.

1.3. 1188 uzņēmumu katalogs Internetā

1188 kataloga projekts paver iespējas iegūt pilnīgu informāciju par Latvijas uzņēmumiem, to darbības veidiem, sniegtajiem pakalpojumiem un piedāvātajām precēm, kā arī uzzināt citu svarīgu informāciju. 1188 uzņēmumu katalogs Internetā piedāvā vienkāršās, detalizētās meklēšanas iespējas, pēc izvēlētā meklēšanas kritērija, objektu meklēšanu kartē vai atlasi pēc darbības veida tematiski vai alfabētiski sakārtotā nozaru katalogā. Sistēmā var meklēt informāciju pēc: uzņēmuma, preces vai pakalpojuma nosaukuma, tālruņu numura, adreses.

1188 uzņēmumu katalogs ir daļēji pārtulkots arī uz krievu un angļu valodām. Pastāv iespēja izmantot krievu vārdus kā meklēšanas parametrus.

Pakāpeniski palielinājās interese par kartes objektu un uzņēmumu meklēšanu Internetā un uzņēmumu katalogs kļuva pieejams vairākos kanālos Internetā.



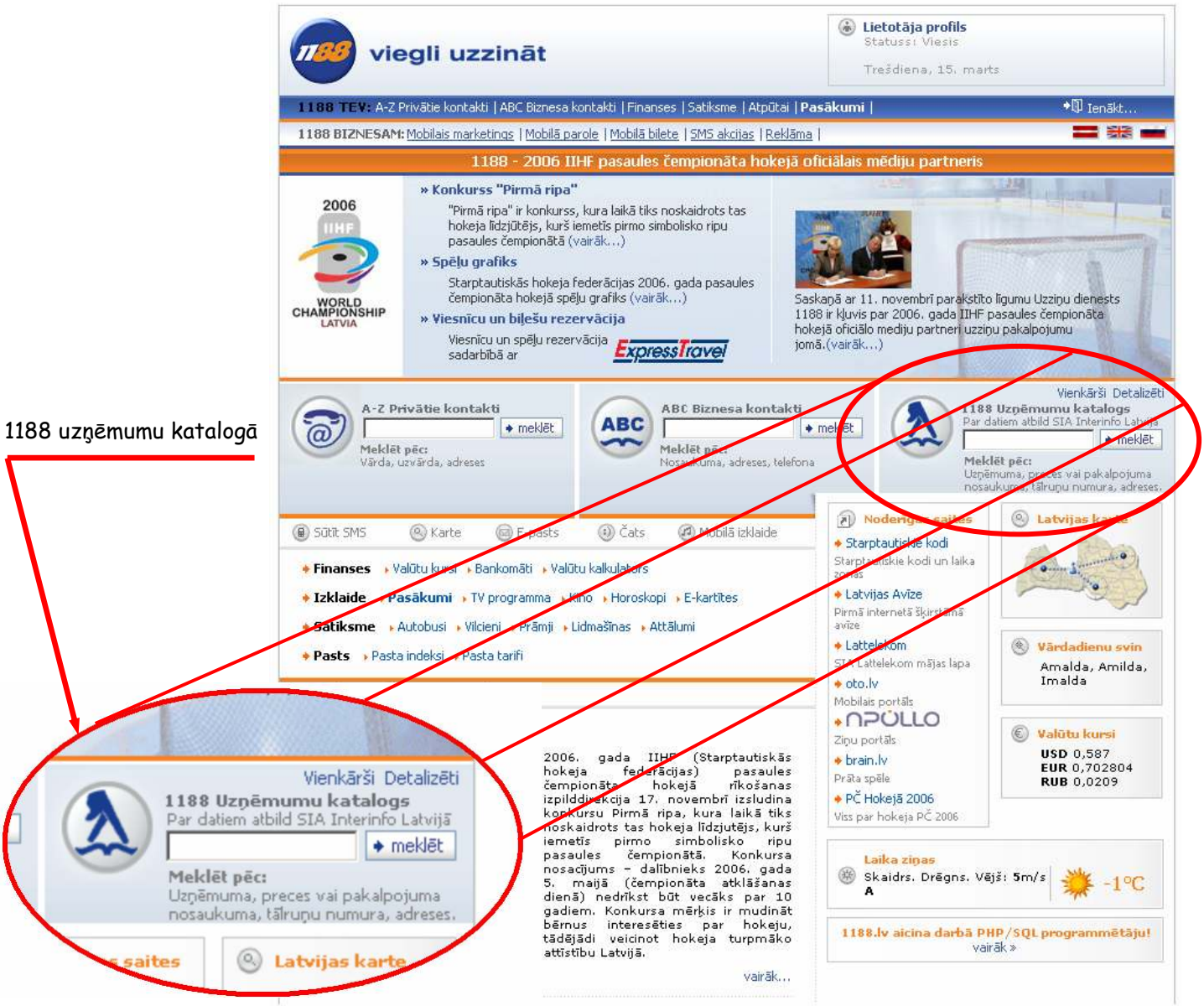
Zīmējums 1. Uzņēmumu kataloga pieejas kanālu vēsture

Zīmējumā 1. ir redzams ka Interneta lappušu skaits, kur var meklēt informāciju par uzņēmumiem 1188 datu bāzē, ar katru gadu palielinās [6].

1.3.1. Interneta kataloga iespējas

Pirmais jautājums ir: kā nokļūt uzņēmumu katalogā. Kā jau tika minēts, tas ir pieejams vairākos portālos Interneta. Kā piemēru, apskatīsim vissenāko pieejas kanālu- www.1188.lv.

Zīmējumā 2. var redzēt, kā Interneta lappusē 1188 var nokļūt uzņēmumu katalogā. Pastāv vairākas iespējas: nospiest uz apļu zīmējumu kreisā pusē, vai uzrakstot meklēšanas vārdu, piemēram, 'Brīvības' un nospiest pogu 'Meklēt', vai arī nospiest uz 'Vienkārši' vai 'Detalizēti', kuri ļauj uzreiz izvēlēt meklēšanas tipu.



Zīmējums 2. www.1188.lv - piekļuve uzņēmumu katalogam

Uzņēmumu katalogs Internetā dalās uz 4 lielām iespējām vai meklēšanas tipiem: vienkāršā, detalizētā, kartes un tūrisms. Atkarībā no apmeklētāja vajadzībām viņš var atrast piemērotu meklēšanas veidu.

Vienkāršā meklēšana ļauj atrast informāciju pa konkrētu nozari. Nozaru sarakstu var sakārtot divos veidos un saņemt vai tematisko nozaru sarakstu vai alfabētisko nozaru sarakstu. Katrai nozarei blakus ir uzrakstīts uzņēmumu skaits. Katra nozare dalās uz apakšnozarēm. Nospiežot uz nozari vai apakšnozari apmeklētājs saņem pilno Latvijas uzņēmumu sarakstu, kas nodarbojas šajā nozarē. Ja apmeklētājs izvēlās alfabētisko nozaru sarakstu, parādās alfabēta burti. Izvēlot kādu no tiem, var redzēt pilnu apakšnozares sarakstu, kuru nosaukumi sākas ar šo burtu, kā arī klientu skaitu, kas darbojas šajā nozarē. Vienkāršas meklēšanas pieprasījuma rezultāts sastāv no uzņēmuma nosaukuma, adreses, tālruņa un faksa numuriem, darbības apakšnozares un papildus informāciju, ja tāda ir pieejama. Papildus informācija var būt: Info lapa ar detalizētāko informāciju, fotogrāfijas, uzņēmuma mājas lapa, e-pasts vai papildus reklāma. Rezultātu tabulu var šķirot pēc uzņēmumu nosaukumiem vai pēc adresēm.

Detalizētā meklēšana ir iespēja meklēt uzņēmumus pēc dažādiem parametriem:

nosaukums, adrese, rajons (var izvēlēties no saraksta: Kurzeme, Latgale, Vidzeme, Zemgale, Daugavpils rajons, Krāslavas rajons, Aizkraukles rajons, Alūksnes rajons, Balvu rajons, Bauskas rajons, Cēsu rajons, Dobeles rajons utt.), pilsēta/pagasts, tālrunis, produkti/pakalpojumi. Ir nepieciešams aizpildīt kaut vienu laukumu. Meklēšanas precizitātei var aizpildīt vairākos laukus vienlaicīgi. Piemēram, ja ievest nosaukuma laukā '1188' un adreses laukā 'Raunas' tad rezultāts sastāv tikai no viena uzņēmuma, bet ja uzrakstīt tikai vienu no šiem laukiem tad uzņēmumu skaits būs lielāks, jo būs atrastas vai nu visas 1188 filiāles, vai nu visi uzņēmumi, kas atrodas Raunas ielā. Rezultāta izskats ir tāds pats, ka iepriekšējā gadījuma- vienkāršā meklēšana.

Nākamā nodaļa ir - Kartes. Pirmā nodaļas iespēja ir meklēt konkrēto ielu vai adresi interaktīvajā kartē. Objekta atrašanas vietas meklēšanu var konkretizēt, izvēlot objekta pilsētu. Ja konkrēta pilsēta nav noradīta, meklēšana notiks visā Latvijā. Gadījumā, ja noradīta iela vai adrese eksistē vairākās pilsētās, parādās atrasto pilsētu saraksts. Nospiežot uz vienu no tiem apmeklētājs var dabūt pilsētas kartes fragmentu ar sameklēto objektu. Otrās iespējas raksturojums ir sekojošs: apmeklētājam ir piedāvāts rajonu, pilsētu un pagastu saraksti. Izvēlot vienu nosaukumu no saraksta, lietotājs var apskatīt vietas apkārtni interaktīvajā kartē.

Visjaunākā no Kartes nodaļas iespējām ir attāluma mērītājs. Apmeklētājs var iezīmēt vairākus punktus pēc kārtas. Kartē paradīsies nogriežņi, kuru virsotnes- apmeklētāja uzzīmēti punkti. Attāluma mērītājs automātiski saskaitīs iegūtā ceļa garumu.

Kartēm ir pieejami vairāki mērogi no 500 kilometriem līdz 160 metriem. Patreiz objektu meklēšana kartē ir pieejama tikai lielās Latvijas pilsētās.

Pēdējais kataloga meklēšanas tips ir Tūrisms, kas ļauj meklēt un apskatīt Latvijas interaktīvajā kartē dažādus dabas objektus.

Sistēmai ir savi trūkumi, kā, piemēram, par vienu no tiem var uzskaitīt informācijas rediģēšanas neiespējamību. Piemēram, ja vienam uzņēmumam ir vairāki telefona numuri un uzņēmuma pārstāvji vēlās, lai 1188 katalogā paradītos cits numurs nekā tur ir piefiksēts, uzņēmuma darbiniekiem nav iespējas pašiem mainīt savu informāciju.

1.4. Papildus pakalpojumi

1188 piedāvāto pakalpojumu skaits ar katru gadu palielinās. Vienlīdzīgi ar uzziņu saņemšanu pa tālruni, tālruņu katalogiem un uzņēmumu katalogu Internetā pastāv arī citas iespējas. Kā, piemēram, tiešais savienojums ar vajadzīgo numuru. Tas nozīmē savienojums ar vēlamās firmas, iestādes vai privātpersonas (jāzina vārds, uzvārds, adrese) sameklēto numuru, izmantojot operatora starpniecību. Šo iespēju var izmantot zvanot uz 1188 no Lattelekom tīkla tālruņiem, taksofoniem, zvanu kartēm vai no LMT mobilajiem tālruņiem.

No 2001.gada 11.decembra Uzziņu dienests 1188 piedāvā pakalpojumu – 1188 mobilā uzziņa. Piezvanot Uzziņu dienestam 1188, var pieprasīt, lai nepieciešamo uzziņu atsūta īsziņas veidā uz mobilo tālruni. Šī iespēja ir pieejama LMT un O-kartes lietotājiem [7].

Nesen parādījās tādi pakalpojumi kā: Preses abonēšana (var abonēt izdevniecību Preses nams, Santa, Sieviete, Rīgas Viļņi, Mūsmājas un citus izdevumus), Atgādinājuma zvans un Taksometra izsaukšana.

2. Uzziņu dienesta 1188 datu bāzes apraksts

Par datu bāzes serveri tiek izmantots Sybase Adaptive Server Enterprise, kas ir viens no trijiem datu bāzes pārvaldības sistēmām, kurus šodien piedāvā Sybase.

Pieaugošā informācijas plūsma mudina uzņēmumus ieguldīt līdzekļus kvalitatīvas programmatūras un tehniskā nodrošinājuma ieviešanā. Palielinoties apstrādājamās informācijas apjomam, paaugstinās sistēmu izveides un uzturēšanas izmaksas, to uzbūve paliek arvien komplicētāka. Vēl vairāk, daļa informācijas uzņēmumā tiek uzturēta savrupos blokos, kuros tā ir it kā iesprostota, ar sarežģītu piekļuvi, un tas neapmierina dinamiskās biznesa prasības. Sybase informācijas pārvaldības produkti palīdz veikt komplicētu informācijas pārvaldi un likvidēt iekšējās barjeras, panākot, ka informācija ir uzticama, noderīga, būtiska un ar ekonomisku pielietojumu.

Runājot par lielo sistēmu būvēšanas nepieciešamību ar simtu un tūkstošu aplikāciju uzturi ar atbilstošiem drošības un apjoma prasībām, Sybase ASE ir optimāls risinājums. Tas ir ekonomiskā relāciju datu bāze ar vienkāršu izmantošanā interfeisu, kas risina pieaugošās prasības datubāzes apjomam un transakciju skaitam, vienlaikus piedāvājot izmaksu ziņā efektīvu datu pārvaldības sistēmu [8].

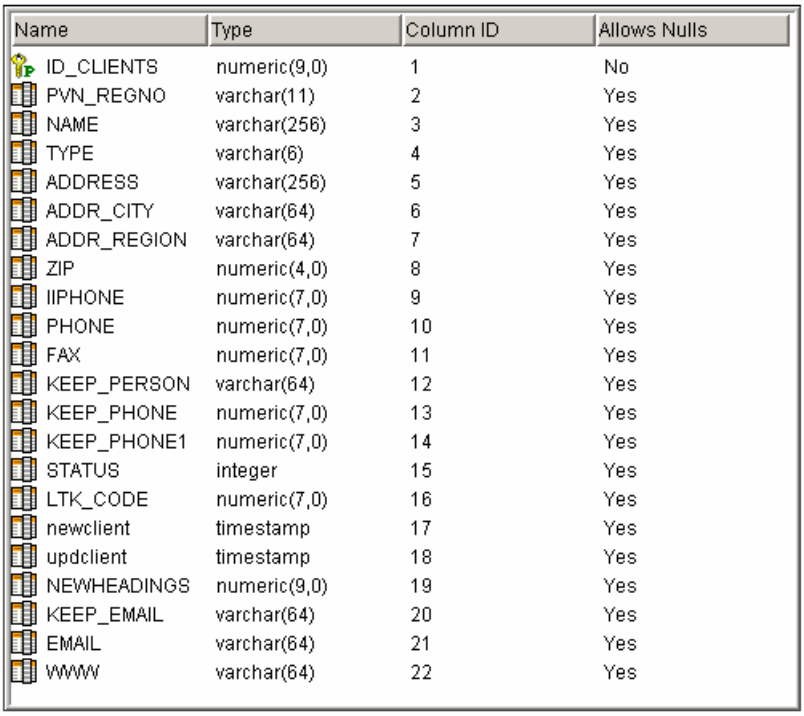
Starp nozīmīgākajām ASE spējām var minēt datu šifrēšanu diska līmenī, viedas partīcijas un jaunu pieprasījumu apstrādes tehnoloģiju, kas ir demonstrējusi ievērojamus veiktspējas uzlabojumus, kā arī atbalstu nestrukturizētu datu apstrādei. ASE ir augstas veiktspējas datubāze, uzticama biznesam kritiskajos lietojumos [9].

1188 uzņēmumu kataloga datu bāzē ir plaša un juridiski pamatota informācija par visiem Latvijā reģistrētiem uzņēmumiem, sabiedriskām organizācijām un ārvalstu uzņēmumu pārstāvniecībām. Datu bāze sastāv no 176 tabulām, kur glabājas informācija par pašiem uzņēmumiem, 1188 katalogiem, karšu objektiem, sadarbības līgumiem, konsultantiem, kas slēdz līgumus, kā arī cita informācija.

Sistēmas izstrādāšanas laikā bija izveidota datu bāzes struktūra, kurai piemīt daža informācijas redundance. Vieni un tie paši dati glabājas paralēli vairākās tabulās. Tas palīdz palielināt meklēšanas ātrumu. Otra šīs pieejas priekšrocība ir datu bāzes drošības palielināšana. Ja, kaut kāda iemesla dēļ, notiks daļēja datu zaudēšana pastāv iespēja atjaunot datu bāzes informāciju.

2.1. Uzņēmumu un adrešu tabulu aplūkošana

Apskatīsim visgalvenāko datu bāzes tabulu- TAB_CLIENTS (sk. Zīmējumu 3.), kas satur informāciju par visiem uzņēmumiem.



Name	Type	Column ID	Allows Nulls
ID_CLIENTS	numeric(9,0)	1	No
PVN_REGNO	varchar(11)	2	Yes
NAME	varchar(256)	3	Yes
TYPE	varchar(6)	4	Yes
ADDRESS	varchar(256)	5	Yes
ADDR_CITY	varchar(64)	6	Yes
ADDR_REGION	varchar(64)	7	Yes
ZIP	numeric(4,0)	8	Yes
IIPHONE	numeric(7,0)	9	Yes
PHONE	numeric(7,0)	10	Yes
FAX	numeric(7,0)	11	Yes
KEEP_PERSON	varchar(64)	12	Yes
KEEP_PHONE	numeric(7,0)	13	Yes
KEEP_PHONE1	numeric(7,0)	14	Yes
STATUS	integer	15	Yes
LTK_CODE	numeric(7,0)	16	Yes
newclient	timestamp	17	Yes
updclient	timestamp	18	Yes
NEWHEADINGS	numeric(9,0)	19	Yes
KEEP_EMAIL	varchar(64)	20	Yes
EMAIL	varchar(64)	21	Yes
WWW	varchar(64)	22	Yes

Zīmējums 3 . TAB_CLIENTS

Zīmējumā ir parādīti visi tabulas lauki ar datu tipiem un kolonu identifikatoriem. Pēdēja kolona nosaka: vai tabula rindiņas lauks var būt tukšs, tas nozīmē: vai viņš ir obligāts aizpildīšanai.

Informācija par TAB_CLIENTS kolonām:

1. *id_clients* - universālais klienta numurs- primārā atslēga
2. *pvn_regno* - uzņēmuma reģistrācijas numurs; reģistrācijas numurs nevar būt primārā atslēga šajā gadījuma, jo viņš var sakrīt diviem dažādiem klientiem, kuri ir viena uzņēmuma filiāles.
3. *name* - uzņēmuma nosaukums
4. *type* - uzņēmuma tips, kas var būt, piemēram, SIA - sabiedrība ar ierobežotu atbildību , AS - akciju sabiedrība un citi
5. *address* - uzņēmuma adrese
6. *addr_city* - pilsēta, kur atrodas uzņēmums
7. *addr_region* - reģions, kur atrodas uzņēmums, piemēram, Rīgas rajons, Liepājas rajons, Mārupes pagasts; daudziem klientiem ir tukšs.
8. *zip* - uzņēmuma pasta indekss
9. *iiphone* - uzņēmuma tālruņa numurs
10. *phone* - otrais uzņēmuma tālruņa numurs, kas tiek izmantots nepieciešamības gadījumā
11. *fax* - uzņēmuma faksa numurs
12. *keep_person* - uzņēmuma darbinieks, kas atbild par reklāmu; lauks ir aizpildīts galvenokārt klientiem, kuriem kādreiz bija līgums ar 1188 dienestu
13. *keep_phone* - iepriekšminēta uzņēmuma darbinieka tālruņa numurs; var būt aizpildīts pat ja *keep_person* lauks ir tukšs, ja, piemēram, uzņēmumā ir reklāmas nodaļa, nevis tikai viens cilvēks, kas atbild par reklāmas līgumiem

14. *keep_phone1* - otrais darbinieka tālruņa numurs, tiek izmantots nepieciešamības gadījumā
15. *status* - klienta statuss; '6' nozīmē, ka klients ir aktīvs, '8' - klients nav aktīvs (klientus nedzēš no tabulas, jo, pirmkārt, viņš var atkal kļūst aktīvs un, otrkārt, nodzēšot klientu, pazūd visa uzņēmuma statistika un vēsture 1188 sistēmā), 118 - klients ir īslaicīgs un eksistē datu bāzē tikai kamēr viņam eksistē līgums
16. *ltk_code* - uzņēmuma identifikators Lattelekom datu bāzē
17. *newclient* - klienta izveidošanas datums datu bāzē
18. *updclient* - klienta pēdējas atjaunošanas datums
19. *newheadings* - ārējā atslēga no tabulas TAB_NEWHEADINGS; uzņēmuma darbības nozares kods
20. *keep_email* - uzņēmuma darbinieka, kas atbild par reklāmu, elektroniskais pasts
21. *email* - uzņēmuma elektroniskais pasts
22. *www* - uzņēmuma lappuse Internetā

Lai veiktu nepieciešamas izmaiņas datu bāzē, būs nepieciešama arī TAB_ADDRESS tabula (sk. Zīmējumu 4.), kas satur informāciju par ielām un mājam. Tabula satur informāciju par Latvijas pilsētu adresēm.

Name	Type	Column ID	Allows Nulls
id_address	numeric(9,0)	1	No
name	char(64)	2	Yes
namenum	char(32)	3	Yes
city	char(32)	4	Yes
x	numeric(9,0)	5	Yes
y	numeric(9,0)	6	Yes
nameeng	char(64)	7	Yes
cityeng	char(32)	8	Yes

Zīmējums 4. TAB_ADDRESS

1. *id_address* - identifikators, primārā atslēga
2. *name* - ielas nosaukums
3. *namenum* - mājas numurs
4. *city* - pilsēta, kur atrodas dota māja
5. *x* – x koordināte
6. *y* – y koordināte
7. *nameeng* - ielas nosaukums angļu valodā
8. *cityeng* - pilsētas nosaukums angļu valodā

Ar *x* un *y* palīdzību var atrast mājas atrašanas vietu interaktīvajā kartē. Informācija par kartēm atrodas TAB_MAPS tabulā, kas satur laukus *x0*, *x1*, *y0*, *y1*, ar kuru palīdzību var noteikt atbilstoša kvadrāta atrašanas vietu kartē. x un y ir unikālas koordinātes visos līmeņos, tātad (x,y) nosaka konkrēto punktu.

2.2. Datu bāzes atjaunošana

Kā jau tika minēts, datu bāzes atjaunošana notiek automātiski. Atjaunošana notiek ar Lattelekom palīdzību. Šim uzņēmumam ir savs uzņēmumu katalogs, kura atjaunošana notiek saskaņā ar Latvijas Republikas uzņēmumu reģistra doto informāciju. Divas reizes nedēļā Lattelekom nosūta datnes, kuri satur informāciju par jaunizveidotiem un aizvērtiem uzņēmumiem, kā arī uzņēmumiem, kuru informācija (piemēram, adrese) tika mainīta.

Failu apstrāde notiek ar speciālo skriptu palīdzību. Pēc apstrādāšanas informācija par izmaiņām nokļūt TAB_LTKIMPORT (sk. Zīmējumu 5.).

Name	Type	Column ID	Allows Nulls
id_ltkimport	numeric(9,0)	1	No
import_code	char(32)	2	Yes
ltk_code	numeric(7,0)	3	Yes
name	char(256)	4	Yes
fullheading	char(6)	5	Yes
address	char(256)	6	Yes
city	char(64)	7	Yes
region	char(64)	8	Yes
zip	numeric(4,0)	9	Yes
phone	numeric(7,0)	10	Yes
loaddate	timestamp	11	Yes
importtype	integer	12	Yes
newheadings	numeric(9,0)	13	Yes
description	char(128)	14	Yes
lur	char(18)	15	Yes

Zīmējums 5. TAB_LTKIMPORT

Šai tabulai ir gan līdzības, gan atšķirības salīdzinājumā ar TAB_CLIENTS. Apskatīsim laukus, kuru nebija uzņēmumu tabulā:

- *id_ltkimport* - identifikators, primārā atslēga
- *import_code* - Lattelekom konkrētas izmaiņas kods
- *fullheading* - uzņēmuma nozares kods Lattelekom datu bāzē (1188 un Lattelekom nozares nosaukumi un kodi atšķiras); nozares atšifrēšana notiek šajā pašā procedūrā, kas apstrādā saņemtus datus
- *loaddate* - faila ielādēšanas datums
- *importtype* – skaitlis, kas nosaka darbību ar ierakstu; 11- nozīmē ieraksta atjaunošana, 12- ieraksta dzēšana (statusa maiņa uz neaktīvu), 13- jauna ieraksta izveidošana
- *description* - ieraksta paskaidrojums
- *lur* - uzņēmuma reģistrācijas numurs

Tabulas apstrāde notiek ar trigera palīdzību, kas izpildās pēc jaunas rindas iestarpināšanas TAB_IMPORT tabulā.

TAB_LTKIMPORT tabulas satura izmaiņas ietekmē TAB_LTKFULLBASE tabulas datus. Šajā tabulā glabājas Lattelekom informācija par uzņēmumiem. TAB_LTKFULLBASE struktūra ir ļoti līdzīga aprakstītai tabulai- TAB_LTKIMPORT.

Starp TAB_LTKIMPORT un TAB_LTKFULLBASE tabulām ir dažas atšķirības: TAB_LTKFULLBASE satur savu identifikatoru un primāro atslēgu- *id_fullbase*, *status* - kas

nosaka uzņēmuma statusu (aktīvs vai neaktīvs), kā arī *loaddate* lauka vietā ir *lastdate* lauks, kas ir ieraksta pēdējās atjaunošanas datums. Šī tabula saistās gan ar TAB_CLIENTS, gan ar TAB_LTKIMPORT. Mainot informāciju TAB_LTKFULLBASE tabulā mainās arī TAB_CLIENTS tabulas dati. Apstrāde notiek trigerī, kas izsaucās pēc TAB_LTKFULLBASE tabulas ierakstu izmaiņām (after update).

3. Uzdevumu formulējumi

Datu bāze ir struktūra, kurā tabulas ir saistītas savā starpā ar noteiktām saitēm, lai nodrošinātu dažādu uzdevumu risināšanu. Mūsdienās datu bāzes sistēmu izmantošana kļūst arvien problemātiskāka. Tas ir saistīts ar to, ka palielinās prasību apjoms tipiskām sistēmām. Jo vairāk informācijas satur datu bāze, jo ilgāk izpildīsies vaicājumi un jo grūtāk būs veikt datu bāzes uzturēšanu [10].

Vissvarīgākie uzņēmumu kataloga uzdevumi ir dot aktuālo informāciju par uzņēmumiem. Tas nozīmē, ka ļoti svarīgi ir savlaicīgi atjaunot datu bāzi, kā arī piedāvāt daudz dažādas papildus iespējas.

3.1. Datu atjaunošanas optimizācija

Datu bāzes atjaunošana un papildināšana ir svarīgs jautājums jebkurā sistēmā. Runājot par uzņēmumu katalogiem Internetā, var teikt, ka regulārā un savlaicīga datu atjaunošana nosaka datu kvalitāti. Sakarā ar daudzām esošām 1188 uzziņu dienesta datu bāzes atjaunošanas problēmām, kā, piemēram, ilgs izpildes laiks, iespējama ierakstu dublikātu veidošana, tika pieņemts lēmums par datu bāzes automātiskās atjaunošanas optimizāciju, lai padarītu to ātrāku un ērtāku.

3.2. Jauno iespēju realizācija

Visjaunākās un visinteresantākās 1188 uzņēmumu kataloga Internetā iespējas ir interaktīvās kartes apskatīšana, kā arī objektu meklēšana tajās. Līdz ar to interaktīvās kartes attīstīšana ieņem visai svarīgu vietu visa uzņēmuma darbībā.

1188 uzziņu dienesta Interneta uzziņu karšu interfeiss ir izstrādāts Macrometia Flash tehnoloģijā, kas dod iespēju vieglāk uztvert mērogu maiņu un pārvietošanas. Interaktīvās kartes ir pieejamas sešos mērogos. Katrs nākamais mērogs ir piecas reizes lielāks. Šobrīd Latvijas kartē ir iespējams atrast kartes fragmentus pēc ielu nosaukumiem vai adresēm. Kartes meklēšana pēc uzņēmumu nosaukumiem nav pieejama, jo viena tāda pieprasījuma izpildīšanai nepieciešams ļoti daudz laika. Tas ir saistīts ar lielo datu apjomu, kā arī dažādo adreses uzrakstīšanas formu problēmām, kas nozīmē, ka pieprasījumā jāpārveido adrešu teksta rindas lai, piemēram, ‘K. Barona’, ‘Barona’ un ‘Krišjāņa Barona’ ielas tika uzskatītas par vienādām. Šo situāciju var izmainīt veicot daļēju datu bāzes pārveidošanu, kas iekļauj uzņēmumu un adrešu tabulu saiti. Tas ļaus uzzināt uzņēmumu atrašanas vietas kartē jeb koordinātes. Tas ir otrais darba uzdevums.

Meklēšanas iespējas nav vienīgais, kas ietekmē cilvēka interesi par interaktīvām kartēm. Ļoti nozīmīga ir dažādu objektu apskatīšana kartēs, piemēram, dabas objektus vai transportu maršrutus. 1188 uzziņu dienests sadarbībā ar SIA Rīgas Satiksme nolēma izstrādāt sistēmu jeb kartes pielikumu, kas ļaus apskatīt Rīgas pilsētas un piepilsētas maršrutus interaktīvajā kartē. Tātad trešais darba uzdevums ir šīs iespējas realizācija no datu bāzes viedokļa, kas iekļauj datu bāzes struktūras ieviešanu (tabulu un relāciju projektēšanu), datu kopēšanu no SIA Rīgas Satiksme datu bāzes un to pārveidošanu nepieciešamā formātā, kā arī tālāko perspektīvu aplūkošanu un to daļēju realizāciju.

4. Izmaiņas datu bāzē un datu struktūrā

Viens no datu bāzes izmaiņu uzdevumiem ir datu bāzes optimizācija. Datu bāzes optimizācija ir datu bāzes pamatparametru optimāla izvēle ņemot vērā atmiņas un cieto disku resursus, procesoru, ievadizvades iespējas, kā arī datu bāzes iekšējus procesus. Datu bāzes optimizācija iekļauj sevī datu bāzes vislabākās (no darba ātruma viedokļa) struktūras, tai skaitā arī tabulu un indeksu, izvēli [11].

Optimizācija ir sarežģīts uzdevums, jo tā prasa pilno sistēmas izpratni. Lai palielinātu sistēmas ātrumu, pirmkārt, bez šaubām, ir nepieciešams orientēties tās konstrukcijā. Turklāt, vajag zināt, kādas funkcijas jāizpilda sistēmai un kādas "šauras vietas" viņai ir.

Zemāk ir minēti visbiežāk sastopamo "šauro vietu" saraksts [12]:

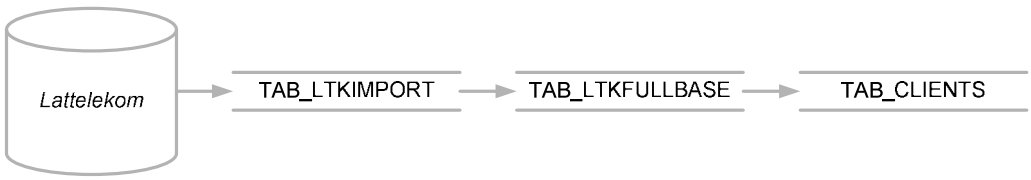
- Datu meklēšana diskā. Lai atrastu kādu datu fragmentu diskā, ir nepieciešams kāds laiks, kuru var būtiski samazināt nomainot jaunākos diskus. Ļoti grūti optimizēt meklēšanu diskā vienai tabulai. Šo optimizāciju var veikt, sadalot datus pa vairākiem diskiem.
- Diska lasīšana/rakstīšana. Datus var nolasīt pēc meklēšanas izpildīšanas, kad ir atrasta atbilstoša pozīcija diskā. Diska lasīšanu/rakstīšanu optimizācija ir vieglāka nekā diska meklēšanas optimizācijas, tāpēc lasīšana var notikt paralēli no vairākiem diskiem.
- Procesora cikli. Novietojot datus pamatatmiņā, lai saņemtu rezultātu ir nepieciešami tos apstrādāt. Mazo tabulu eksistence- ir viens no visbiežāk sastopamiem limitējošiem faktoriem.

Jebkura programmēšanas valoda piedāvā vairākus viena uzdevuma risināšanas paņēmienus. Protams, salīdzinot ar citiem, daži no tiem dot labāko produktivitāti. Datu bāzes optimizācijai jāpievērš lielo uzmanību it īpaši lielo datu apjoma gadījumā, jo korekta optimizācija var būtiski palielināt pat vienkārša pieprasījuma izpildīšanas ātrumu, bet nevar aizmirst, ka datu bāzes optimizācijas iespējas ir atkarīgas ne tikai no programmēšanas valodas un tās īpašībām, bet arī no datu bāzes struktūras [13].

4.1. Datu atjaunošanas optimizācija

Datu bāzes atjaunošanas procesā atjaunojas trīs tabulas: TAB_LTKIMPORT, TAB_LTKFULLBASE un TAB_CLIENTS. Katra no tiem satur vairākus tūkstošus ierakstus.

Zīmējumā 6. ir redzams kā no Lattelekom datu bāzes informācija caur tabulām ar trigeru palīdzību nonāk TAB_CLIENTS. Uz viena faila apstrādi tērēs vairākas stundas. Vissarežģītākā atjaunošanas daļa ir tieši tabulu satura maiņa, jo konkrētā ieraksta meklēšanai un apstrādei vienā no tabulām ir nepieciešams ļoti daudz laika.



Zīmējums 6. Datu atjaunošanas shēma

Datu atjaunošanas optimizāciju var veikt dažādi. Viens no metodēm ir: saīsināt datu pārraides ceļu un atjaunojot TAB_LTKIMPORT tabulas datus, mainīt uzreiz TAB_CLIENTS datus. Šajā gadījumā datu atjaunošana aizņems mazāk laiku, bet rādās jaunas problēmas. TAB_LTKFULLBASE ir vienīgā tabula, kur informācija glābās tādā pašā veidā kā Lattelekom datu bāze. Tas nozīmē, ka, ja atklāsies kļūda datos, tad nebūs īsti zināms vai šī kļūda ir Lattelekom kļūda, vai 1188 Uzziņu kataloga kļūda un var rādīties papildus uzdevumi, kā, piemēram, būs grūti noķert, kur un kad veidojas kļūda un kā to izlabot.

Kā otro optimizācijas variantu var mēģināt vienkāršot TAB_LTKFULLBASE tabulu un nodzēst no šīs tabulas visus neaktīvus klientus, jeb uzņēmumus, kuri uz doto brīdi ir slēgti. Šinī gadījumā arī nebūs pilna priekšstata par Lattelekom datu bāzi, bet tas nebūs visgalvenākais trūkums. Bieži sanāk tā, ka neaktīvs klients pēc kāda laika atkal kļūst aktīvs. Nebūtu problēmas izveidot to no jauna, bet jāatceras, ka TAB_CLIENTS satur lauku *ltk_code*, kas savieno šīs divas tabulas, un ja gadījumā kāda uzņēmuma *status* no neaktīva nomainīsies uz aktīvo, veidosies uzņēmumu dublikāti.

4.1.1. Izvēlētās metodes apraksts

Izvēlētā optimizācijas metode sastāv no vairākiem etapiem, kas maina ne tikai tabulu struktūru, bet arī trigerus un procedūras, kas apstrādā šis tabulas.

Pirmais solis ir *importtype* lauka nozīmes maiņa. Izmaiņas būtība ir: pēc Lattelekom failu apstrādes ieraksti tiks saglabāti TAB_LTKIMPORT tabulā ar tipiem: 1, 2, 3, bet tieši datu bāzes un klientu tabulas atjaunošana notiks, kad pie tipa lauka pieskaita 10. Kā rezultātā *importtype* vērtība apstrādātam TAB_IMPORT ierakstam būs vienāda ar 11, 12 vai 13. Ja kāda tabulas rindiņa vēl netika apstrādāta *importtype* lauka vērtība paliks mazāka vai vienāda ar 3. Tas ir izdarīts ērtībai. Tagad datus var atjaunot pēc izvēles jebkurā laikā, piemēram, pa nakti, vai brīvdienās, kad apmeklētāju skaits samazinās, atlasot tikai tabulas nepieciešamus ierakstus.

Darba gaitā tika nodzēsts iepriekšējais TAB_LTKIMPORT trigeris un izveidots jauns, kas fiksē izmaiņas tabulā (after update). Atkarībā no ieraksta tipa, trigeris izpilda dažādas darbības. Ja darbība netika izpildīta, trigeris maina *importtype* lauka vērtību uz iepriekšējo. Trigera tekstu var apskatīties pielikuma 1.punktā - TAB_LTKIMPORT update.

Otrais optimizācijas solis ir jaunas kolonas izveidošana tabulā TAB_LTKIMPORT. Jaunas kolonas nosaukums ir *importerror*. Lauka vērtība nav tukša tikai gadījumos, kad atbilstošais tabulas ieraksts kaut kāda iemesla dēļ netika apstrādāts, tātad *importtype* palika neizmainīts. *Importerror* uzdevums ir atklāt kļūdas atjaunošanas procesā. Lauka vērtības var būt, piemēram, sekojošas: 'Company name not exist' vai 'LTK code is 0 or not recognized'. Pirmā ir iespējama ja *name* lauks (uzņēmuma nosaukums) nav noradīts un ir tukšs; otrais- ja ir problēmas ar *ltk_code* lauku-identifikatoru Lattelekom datu bāzē.

Trešais etaps ir pēdējas datu atjaunošanas soļa optimizācija: TAB_LTKFULLBASE tabulas izmaiņu apstrādāšana. Optimizācijas būtība: jaunas tabulas izveidošana, kur glabāt tikai to uzņēmumu informāciju, kuri uz doto momenta brīdi ir aktīvi. Šinī gadījumā visa informācija par uzņēmumiem nebūs sabojāta, kā arī atšķirībā no iepriekšējām metodēm būs pieejama to vēsture un statistika.

Jaunizveidota TAB_LTKCLIENTS tabula ir ļoti līdzīga TAB_LTKFULLBASE tabulai. TAB_LTKCLIENTS tabulā ir savs identifikators, kas kalpo par primāro atslēgu, bet galvenā atšķirība ar visu uzņēmumu tabulu ir *status* lauka trūkums, jo visi klienti ir aktīvi, tātad statuss viņiem ir vienāds. Tabula satur lauku *ltk_code*, kas saista šo tabulu gan ar TAB_LTKFULLBASE, gan ar TAB_CLIENTS.

Zīmējums 7. Datu atjaunošanas shēma pēc optimizācijas

Zīmējumā 7. ir paradīts kā notiek datu atjaunošana pēc optimizācijas. Ir redzams, ka atšķirībā no iepriekšējas shēmas soļu skaits ir lielāks. Tas ļauj samazināt atjaunošanai tērētu laiku, jo TAB_LTKCLIENTS tabula satur gandrīz par piecpadsmit tūkstošiem mazāk ierakstu nekā TAB_LTKFULLBASE tabula, tātad būtiski samazinās konkrēta ieraksta meklēšanas laiks klientu tabulā.

Darba laikā tika izlabots trigeru teksts, kas izpildās pēc TAB_LTKFULLBASE atjaunošanas, kā arī pēc rindas iestarpināšanas tabulā. Trigeri atjauno TAB_LTKCLIENTS, kur *ltk_code* sakrīt ar vajadzīgo kodu TAB_LTKFULLBASE tabulā, bet ja tādu nav – uztaisa jauno ierakstu. TAB_LTKFULLBASE tabulas atjaunošanas un rindas iestarpināšanas trigeru tekstus var apskatīt pielikumā (sk. 2. un 3. punktu- TAB_LTKFULLBASE insert, TAB_LTKFULLBASE update). Pagaidām TAB_CLIENTS tabulas dati netika mainīti.

Tika uzrakstīta procedūra *ltkcheckclient*, kas izsaucās trigerī pēc datu izmaiņām TAB_LTKCLIENTS tabulā:

```
call internet.ltkcheckclient(new_fb.ltk_code);
```

Kur internet ir procedūras izveidotājs; new_fb.ltk_code- parametrs. Kā parametru procedūra saņem izmainīta vai jaunizveidota ieraksta identifikatoru - *ltk_code*. Procedūra *ltkcheckclient* izpildās visiem jaunajiem un atjaunotiem TAB_LTKFULLBASE ierakstiem.

Ltkcheckclient atkarībā no ieraksta eksistences TAB_CLIENTS tabulā, kur Lattelekom kods sakrīt ar parametra vērtību, izpilda dažādas darbības:

- ✓ Ja nav ieraksta- iestarpina jauno ierakstu;
- ✓ Ja ir viens ieraksts ar neaktīvo statusu - atjauno ieraksta laukus atbilstoši TAB_LTKCLIENTS tabulas datiem, un maina *status* uz 6, klients kļūst atkal aktīvs;
- ✓ Ja ir viens ieraksts ar aktīvo statusu- maina ieraksta lauku datus, par pamati ņemot TAB_LTKCLIENTS tabulas datus;
- ✓ Ja ir vairāki ieraksti - atjauno datus uzņēmumam, kuram eksistē līgumi, bet ja starp ierakstiem tādu nav, atjauno ierakstu ar minimālo identifikatoru; pārējiem klientiem ar šo pašu kodu *status* nomaina uz 8 (neaktīvo) un *ltk_code* nomaina uz NULL (tukšums);

Procedūra ne tikai garantē kļūdu neveidošanos, kas saistās ar dublikātiem un neaktīviem klientiem, bet arī labo iepriekš izdarītas kļūdas (ja eksistē vairāki ieraksti ar vienādo *ltk_code* lauka vērtību, kas pēc būtības ir unikāls kods un nevar atkārtoties nekādā gadījumā). Pie tam palielinājās ne tikai konkrēta soļa izpildīšanas ātrums, bet arī visa atjaunošanās procesa ātrums. *Ltkcheckclient* procedūras pilno tekstu sk. pielikuma 4.punktā- LTKCHECKCLIENT procedure.

Izdarītie soļi ne tikai ļoti būtiski samazināja atjaunošanas izpildes laiku, bet arī izdarīja šo procesu ērtāk. Parādījās iespēja izvēlēties laiku, kas labāk piemērots datu bāzes atjaunošanai, kad

apmeklētāju skaits nav liels. Samazinājās kļūdu veidošanās risks, kā arī pievienota automātiskā veco kļūdu noķeršana un labošana.

4.2. Izmaiņas tabulu struktūrā

Interneta lappuses apmeklētāju skaits, pirmkārt, ir atkarīgs no piedāvātiem pakalpojumiem un iespējām. Runājot par uzņēmumu katalogiem Internetā viens no svarīgākiem uzdevumiem ir dot pēc iespējas pilno priekšstatu par visiem uzņēmumiem. Uzņēmuma adresi, telefona numuri un darbības nozari var uzzināt zvanot pa tālruni. Uzņēmumu kataloga iespējām vajadzētu būt plašākām, kā piemēru var minēt uzņēmuma meklēšanu Latvijas interaktīvajā kartē.

Kā jau tika minēts, šīs iespējas pievienošana ir iespējama arī bez izmaiņām datu bāzē. Šajā gadījumā, saņemot uzņēmuma nosaukumu vajadzētu ne tikai atrast derīgus ierakstus uzņēmumu tabulā, bet arī saistīt atbilstošo uzņēmumu adreses ar konkrētām vietām interaktīvajā kartē. To var izdarīt tikai ar adreses tabulas palīdzību, tātad jāapskata visus ierakstus šajā tabulā. Viena tāda pieprasījuma veikšanai vajadzētu vismaz piecas minūtes. Datu bāzes optimizācija ļaus vairākkārt samazināt šo laiku.

4.2.1. Adrešu tabulas pārveidošana

Vienu adresi var uzrakstīt ļoti dažādi un par to nevar aizmirst, jo šī problēma ir viena no galvenajām problēmām adrešu salīdzināšanā. Piemēram, ‘K.Valdemāra’, ‘Krišjāņa Valdemāra’ un vienkārši ‘Valdemāra’, dators uzskata par trim dažādām ielām, kā arī pastāv iespēja, ka cilvēks, kas ievadīja datus uzrakstīja to ar kļūdām, piemēram aizmirsa ielikt garumzīmi. Ņemot vērā šīs kļūdas un uzrakstīšanas īpatnības optimizācijas pirmais solis: pārveidot adreses speciālā formā, lai cilvēka kļūdas neietekmēja uz uzdevuma risināšanu.

Visas adreses glabājas TAB_ADDRESS tabulā un optimizācijas primārais uzdevums ir izmainīt tieši šo tabulu. Lai katru adresi saistīt ar atbilstošo teksta rindu, kas veidosies adreses pārveidošanas procesā, tabulai tika pievienota jauna kolona ar nosaukumu *easyname*. Nākošais uzdevums- izdomāt kādas pārveidošanas ar adresi jāizdara. Kā jau tika minēts lielas grūtības izraisa pareizrakstības kļūdas, tātad no sākuma vajag pārveidot ielas nosaukumu uz angļu alfabēta burtiem, tas nozīmē, ka ‘ā’ vietā jādabū ‘a’, ‘ž’ vietā - ‘z’. Tas attiecas uz visiem latviešu burtiem, kuriem neeksistē analoga angļu alfabētā. Piemēram, ‘Lāčplēša iela’ jāpārveido par ‘Lacplesa iela’.

Ļoti bieži, rakstot adresi, cilvēki neraksta vārdu ‘iela’ un nav svarīgi vai adrese satur šo vārdu vai nē. Tātad nākošais pārveidošanas solis ir vārda ‘iela’ dzēšana no adresēm. Kad pirmie soļi ir izdarītie var pāriet pie nākošiem, kuru nozīme ir lielāka. Atgriežoties pie jautājumam, kas saistās ar Krišjāņa Valdemāra ielu un tās dažādiem rakstīšanas veidiem, var teikt, ka šī iela nav vienīga, kuras nosaukums sastāv no vairākiem vārdiem, vienu no kuriem var nerakstīt, pie tam ļoti svarīgi, ka nerakstīt var tikai pirmo nosaukuma vārdu, jo citā gadījumā adreses pārveidošanas rezultāts nebūs unikāls. Jāizdomā skriptu, kas pārbauda, vai pēc iepriekšējiem pārveidošanas soļiem paliek vairāk par vienu vārdu. Ja nosacījums izpildās, tad pirmo nosaukuma vārdu varam neņemt vērā un pārveidošanas procesā dzēst ārā. Tomēr ir ļoti daudz izņēmumu, jo, piemēram, nodzēšot ‘Mazā Krasta’ ielas nosaukuma pirmo vārdu sanāk vārds ‘Krasta’, kas pēc savas būtības ir pavisam cita iela. Šādu gadījumu ir ļoti daudz, bet ir iespējams atrast visus tādus nosaukuma pirmos vārdus, kurus nevar dzēst un pārveidot tikai tos nosaukumus, kas nesatur vārdu no šī saraksta. Problēmas var rasties arī ar otro vārdu, ja tas ir viens no vārdiem ‘prospekts’, ‘laukums’, ‘aleja’, ‘gatve’ un tā tālāk. Ja šinī gadījumā nodzēš tekošā rezultāta pirmo vārdu, tad rezultāts pēc soļa izpildes būs nepareizs. Šīs problēmas risinājums ir līdzīgs iepriekšējam. Var atrast līdzīgo vārdu sarakstu un ja adrese satur kādu no šī saraksta vārdiem tad iepriekšējo nevaram dzēst.

Pēc iepriekšējiem pārveidojumiem paliek tikai nodzēst rezultātā liekas atstarpes un atzīmēt ar simboliem ielas šifrējuma sākumu un beigās, kā arī pievienot mājas numuru. Beigu rezultātu

jāraksta TAB_ADDRESS tabulas *easyname* laukā. Pēc visu iepriekšējo soļu izpildīšanas katrai adresei būs savs unikālais pieraksts, kas kopā ar pilsētas lauka vērtību nosaka konkrēto māju un punktu Latvijas kartē. Tātad pirmais optimizācijas etaps ir adrešu pārveidošanas procedūras izstrādāšana.

4.2.1.1. Teksta pārveidošanas funkcija

Darba pirmajā posmā ir izstrādāta *normalizetxt* funkcija, kuras mērķis ir teksta rindas normalizācija- teksta pārveidošana. Tas nozīmē, ka funkcija pārveido latviešu burtus par līdzīgiem angļu alfabēta burtiem, visus rindas lielos burtus pārtaisa par maziem un atstarpes rindā nomaina uz punktiem (‘.’). Notiek arī pieturzīmju apstrādāšana, kuras rezultātā daži no tiem arī mainās uz punktiem.

Informāciju par simboliem funkcija saņem no TAB_IDEFAULT tabulas, kas satur tādus laukus kā *nt_novalidchars*, *ltchars* un *engchars*. Šo lauku saturs ir:

- ✓ *nt_novalidchars*:

'`~!@#$%^*()_+={[}]|\:;"<,> ?/|\

- ✓ *latchars*:

ēŗūīōāšģķļžčņĒŖŪĪŌĀŠĢĶĻŽČŅQWERTYUIOPASDFGHJKLZXCVBNM.&-

- ✓ *engchars*:

eruioasgklzcneruioasgklzcnqwertyuiopasdfghjklzxcvbnm.&-

Pirmā lauka saturs ir ‘nederīgie' simboli, kurus funkcija nomaina uz punktiem. Otrais lauks ir latviešu alfabēta burti, kuru nav angļu valodā, kā arī visi lielie burti. Trešais lauks sastāv tikai no angļu burtiem.

Funkcijas darbojas sekojoši: nolasa vienu simbolu no teksta rindas un pārbauda vai šī simbola pozīcija ir lielāka par nulli ‘nederīgo’ simbolu rindā. Ja atbilde ir pozitīva, tad šo simbolu maina uz punktu, ja nē, tad aprēķinā simbola pozīciju *latchars* laukā. Ja pozīcija ir lielāka par nulli, tad atrodam simbolu *engchars* rindā, kura pozīcija sakrīt ar dota simbolu pozīciju *latchars* laukā. Pretējā gadījuma simbols nemainās. Rezultējošo simbolu pievieno jaunizveidotam teksta mainīgam. Pēc visu šo soļu izpildīšanas funkcija nolasa rindas otro simbolu un turpina pārbaudīt simbola piederību *nt_novalidchars* vai *latchars* laukam un pievieno saņemto rezultātu šim pašam teksta mainīgam. Funkcija turpina simbolu pārveidošanu kamēr teksta rinda nav beidzies.

Funkcijas fragments kas pārbauda vai simbols pieder *latchars* laukam un atkarība no tā maina to uz citu vai atstāj tādu pati izskatās sekojoši:

```
set @kk=charindex(@ch,@lchars);
set @kk=isnull(@kk,0);
if @kk > 0 then
  set ch=substring(@echars,@kk,1)
end if;
set @ntwrd=@ntwrd+@ch;
```

Kur:

kk – integer tipa mainīgais, kura vērtība ir vienāda ar simbolu pozīciju latviešu burtos;

ch – nolasītais rindas simbols;

lchars – teksta rinda ar latviešu alfabēta burtiem, kuru nav angļu valodā, kā arī visiem lieliem burtiem;

echars – teksta rinda ar atbilstošiem angļu alfabēta burtiem;

ntwrd – teksta mainīgais, kur glabājas tekošais rezultāts.

Pēc visu simbolu apstrādāšanas, funkcija nomaina divu punktus ('..') ar vienu, ja tādi gadījumi ir un atgriež rezultējošu teksta rindu. Ja izpildīt šo funkciju, piemēram, tekstam 'Krišjāņa Valdemāra iela' rezultāts ir: 'krisjana.valdemara.iela'. *Normalizetext* funkcijas tekstu var apskatīt pielikuma 5.punktā – NORMALIZETEXT function.

4.2.1.2. Adrešu speciālās formas veidošana

Adreasyname procedūras mērķis ir TAB_ADDRESS tabulas pārveidošana. Par pārveidošanu šajā gadījuma uzskatās *easyname* lauka aizpildīšanas visiem tabulas ierakstiem. Šī lauka vērtība ir atkarīga no diviem laukiem: *name* (ielas nosaukums) un *namenum* (mājas numurs).

Procedūra darbojas pēc sekojoša algoritma:

Nolasa pirmā ieraksta *name* lauku, uzreiz nomainot vārdu 'iela' ar tukšo vieto, jo, kā jau tika minēts, šis vārds neietekmē uz adreses speciālo formu. Nākošais solis ir teksta rindas apstrādāšana ar *normalizetxt* funkciju, tātad visi lielie burti nomainās uz maziem, latviešu burti, kuriem nav analogu angļu alfabētā, nomainās uz angļu burtiem.

Tālāk seko visdarbietilpīgākā procedūras daļa, kuras uzdevums ir noteikt, vai tekošais rezultāts pēc funkcijas izpildīšanas sastāv no vairākiem vārdiem, un ja tā ir, tad nolemt vai pirmo adreses vārdu var dzēst ārā, vai arī tas ir obligāts. Pie sliktiem vārdiem, kurus nevar dzēst speciālā adreses formā, pieder tādi vārdi kā, piemēram, 'Mazā' vai 'Jaunā'. Procedūra nolasa teksta rindas pirmo vārdu un vienkārši pārbauda vai šis vārds pieder sarakstam vai nē. Problēmas var sanākt arī ar otro vārdu, ja tas ir viens no vārdiem 'prospekts', 'laukums', 'aleja', 'gatve' un tā tālāk. Nolasot pirmo vārdu procedūra pārbauda vai atlikušais vārds pieder otrajam sarkstam. Ja atbilde ir pozitīva pirmo vārdu nevar dzēst. Atkarībā no pārbaužu rezultāta procedūra atstāj teksta mainīgo vērtību tādu pati, kāda bija pirms pēdējiem soļiem, vai maina, nodzēšot pirmo vārdu.

Pēdējais apstrādāšanas etaps ir mājas numura pievienošana, simbola '.' pievienošanas teksta mainīga sākumā un beigās, kā arī mainīga vērtības ierakstīšana TAB_ADDRESS tabulā. Pirms mājas numura pievienošanas notiek tā apstrāde ar *normalizetxt* funkciju, jo mājas numurs var saturēt 'sliktus' pieturas zīmes, kas jāpārveido par punktiem, piemēram, mājas numurs var būt vienāds ar '12/78' vai '13 kab.17'.

Procedūra pēc kārtas apstrādā visus TAB_ADDRESS ierakstus, pievienojot katram no tiem speciālo formu. Procedūras tekstu var apskatīt pielikumā (sk. 6. punktu – ADREASYNAME procedure).

name	namenum	easyname
Saules aleja	1	.saules.aleja.1.
Siguldas prospekts	10	.siguldas.prospekts.10.
Augusta Deglava iela	120	.deglava.120.
Krišjāņa Valdemāra iela	151A	.valdemara.151a.
L.Paegles iela	16	.paegles.16.
Maza Dzirnavu iela	2	.maza.dzirnavu.2.
Tīnūžu iela	3	.tinuzu.3.
Kartupeļu iela	47	.kartupelu.47.
Krasta iela	5	.krasta.5.
Mazā Krasta iela	5	.maza.krasta.5.
Ezera iela	53/55	.ezera.53.55.
Jomas iela	66 lit.2	.jomas.66.lit.2.

Tabula 1. TAB_ADDRESS tabulas fragments pēc adreasyname procedūras izpildes

Tabulā 1. ir paradīts TAB_ADDRESS tabulas fragments: dažu ierakstu *name*, *namenum* un *easyname* lauku saturs. *Easyname* lauka saturs atbilst sākuma uzdevumam. Speciāla forma nesatur vārda ‘iela’, sastāv tikai no maziem angļu alfabēta burtiem, cipariem un punktiem. Pie tam dažos gadījumos speciāla forma ir saīsināta- tas nozīmē, ka tā var saturēt mazāk vārdu nekā sākotnēja adrese, jo šiem vārdiem nav lielas nozīmes un tos var nerakstīt.

Ļoti svarīgs jautājums ir lauka *easyname* unikalitāte. Tā kā procedūra nodzēš tikai tos pirmos vārdus, kas neietekmē uz ielas unikalitāti, sanāk, ka ņemot vērā *easyname* un *city* lauku saturu- katru ierakstu var viennozīmīgi identificēt. Pilsētas (*city*) lauks ir svarīgs jo dažādās pilsētās var būt ielas ar vieno un to pašu nosaukumu.

Pirmais pārveidošanas solis ir izpildīts. Nākošais uzdevums- uzņēmumu tabulas pārveidošana.

4.2.2. Uzņēmumu tabulas pārveidošana

Lai viennozīmīgi piesaistīt katrai TAB_CLIENTS rindai vienu konkrētu ierakstu TAB_ADDRESS tabulā, jāveic gandrīz tādas pašas izmaiņas arī uzņēmumu tabulā. Adrešu tabulas pārveidošanas laikā tika ņemti vērā dažādi adrešu pierakstu veidi, tātad uzņēmumu tabulas pārveidošana ir vieglāka, jo nav vajadzības skatīties kādus ielas nosaukumu vārdus var dzēst un kādus nevar.

TAB_CLIENTS tabulai tika pievienoti sekojošie lauki: *specaddr*, *addrid*, *x*, *y*. *Specaddr* lauks ir nepieciešams lai glabāt adreses speciālo formu. Speciālas formas raksturojums ir līdzīgs adrešu tabulai. Forma var saturēt tikai mazus angļu alfabēta burtus. Pieraksta sākumā un beigās jābūt punkti, kā arī visas atstarpes un gandrīz visas pieturas zīmes jānomaina uz punktiem. Lauks *addrid* ir nepieciešams lai saistītu uzņēmumu un adrešu tabulas, tātad tas ir adreses identifikators TAB_ADDRESS tabulā. *x* un *y* lauki ir atbilstoša ieraksta koordinātes. Tātad, pēc tabulas pārveidošanas, nolasot klientu uzņēmumu tabulā, būs iespējams uzzināt tā atrašanas vietas atbilstošu punktu interaktīvajā kartē (ar *x* un *y* palīdzību). *Addrid* lauks ir nepieciešams lai nodrošināt datu korektību.

Pārveidošanas primārais uzdevums ir: katram tabulas ierakstam jāatrod adreses speciālo formu un jāieraksta to *specaddr* laukā. Šim laukam kopā ar pilsētas (*city*) lauku jāatbilst konkrētai vietai Latvijā, kā arī vienam punktam interaktīvajā kartē. Tā kā ielas vārds un mājas numurs šajā

tabulā glabājas vienā un tā pašā laukā, kā arī nav jārūpējas par to kādus vārdus var atstāt un kādus nē, speciālas adreses formas lauku var aizpildīt vienkārši, izpildot vaicājumu:

```
update admin.tab_clients set
specaddr = ('.' ||
  trim(internet.normalizetxt(replace(address,' iela',' '))) || '.');
```

Pēc atjaunošanas visiem TAB_CLIENTS tabulas ierakstiem *specaddr* lauks tiks aizpildīts ar adrešu speciālām formām.

Nākošais un pats galvenais pārveidošanas jautājums ir adrešu un uzņēmumu tabulu saite, tas nozīmē *addrid*, *x*, *y* lauku aizpildīšana. Tas nozīmē katram uzņēmumam jāpiesaista konkrētu adresi, jeb noteikto punkti interaktīvajā kartē. Šī uzdevuma risināšanai tika uzrakstīta procedūra.

4.2.2.1. Tabulas saite ar adrešu tabulu

Lai izveidotu saiti starp uzņēmumu un adrešu tabulām tika uzrakstīta procedūra- *addressupdate*. Procedūras mērķis ir TAB_CLIENTS tabulas *addrid* lauka aizpildīšana.

Addressupdate saņem divus numeric(9) tipa parametrus *addrf* un *addrt*. Parametru nozīme ir sākotnēja un beigu ieraksta identifikatori adrešu tabulā. *Addrf* ir ieraksta minimāls identifikators, no kura procedūra sāk apstrādāt tabulas datus, pie tam ierakstu tieši ar šo identifikatoru var nebūt tabulā, šajā gadījumā datu apstrāde sāksies ar pirmo ierakstu, kur identifikators ir lielāks par doto. *Addrt* ir pēdējais identifikators, līdz kuram procedūrai jāizpildās. Izveidotas procedūras izsaukšanas piemērs:

```
call internet.addressupdate(2000, 2500);
```

Kur ‘internet’ ir lietotāja vārds, kas uzskatās par procedūras autoru vai izveidotāju. Procedūra izpildīsies adrešu tabulas ierakstiem, kuru unikāls identifikators ir lielāks vai vienāds ar 2000 un mazāks vai vienāds ar 2500.

Parametri ir izveidoti ērtībai, jo gan TAB_ADDRESS, gan TAB_CLIENTS tabulas satur ļoti daudz ierakstu. Procedūras izpildes laiks ir tieši saistīts ar apstrādājamo ierakstu skaitu. Lai netraucētu datu bāzes darbībai, labāk izpildīt procedūru nevis visiem TAB_ADDRESS tabulas ierakstiem uzreiz, bet sadalot tos pa daļām.

Procedūra ar kursora palīdzību apstrādā TAB_ADDRESS ierakstus pēc kārtas. Nolasot adreses speciālo formu uz teksta mainīgo *nm* un pilsētu uz *ct* mainīgu, *addressupdate* pārbauda vai TAB_CLIENTS tabulā eksistē ieraksti ar *speccadr* lauku, kas satur mainīga *nm* vērtību. Pie tam svarīgi lai sakrītu arī pilsētas lauki, jo vienīgi adreses speciāla forma nav unikāls identifikators. Klientu skaita aprēķināšana notiek sekojoši:

```
select count(*) into acn from admin.tab_clients where
  (specaddr like '%' || nm || '%') and city = ct;
```

Pēc vaicājuma izpildes kopējais uzņēmumu skaits, kuru adreses speciālā formā sakrīt ar vajadzīgo TAB_ADDRESS tabulas ieraksta speciālo formu, tiek ierakstīts uz *acn* mainīgo. Ja *acn* ir 0 tad procedūra nolasa nākošo adrešu tabulas ierakstu un apstrādā tapāt kā iepriekšējo. Ja *acn* ir lielāks par 0, tad jāpārbauda vai adrese patiešām sakrīt ar vajadzīgo TAB_ADDRESS ieraksti. Grūtības var būt, ja, piemēram, uzņēmuma adrese ir Mazā Krasta 5. Šo adresi var uztvert arī par

Krasta 5, tātad jāpārbauda uzņēmuma adreses pirmo vārdu. Vai šis vārds nav viens no ‘sliktiem’, kuru skaitā ir ‘Mazā’, ‘Jaunā’ un citi. Ja pirmais vārds ir viens no saraksta, tad *nm* mainīga kreisa robeža ir stingra. Šis paņēmiens ļauj izvairīties no kļūdām un nesajaukt tādus gadījumus, kā, piemēram, ‘Dzirnavu’ un ‘Maza Dzirnavu’.

Ja uzņēmuma adreses speciālas formas pirmais vārds nepieder ‘slikto’ vārdu sarakstam, tad atjaunošanas kritērijs ir tāds pats kā iepriekšminētam pieprasījumam. Tad mainīga *nm* vērtībai pa kreisi var pievienot jebkādu tekstu:

```
set nm='%' || nm;
```

Atjaunošana abos gadījumos notiek sekojoši:

```
update admin.tab_clients set addrid = ida,x = xx,y = yy
  where specaddr like nm || '%'  and city = ct;
```

Kur *ida* ir adrešu tabulas identifikators, *x* un *y* ir atbilstoša ieraksta koordinātes šajā pašā tabulā. Pēc procedūras izpildes visiem klientiem uzņēmumu tabulā, kuru adrese sakrīt ar vienu no apstrādātiem TAB_ADRESS tabulas ierakstiem, *addrid* kļūst vienāds ar adrešu tabulas ierakstu identifikatoru. Procedūras pilno tekstu sk. pielikuma 7.punktā- ADDRESSUPDATE procedure.

Pēc procedūras izpildes *addrid*, *x* un *y* lauki uzņēmumu tabulā vairākiem ierakstiem ir aizpildītie ar skaitļiem. Procentuāli apmēram 82% no TAB_CLIENTS ierakstiem šie lauki atšķiras no null, tātad nav tukši. Šo skaitli ļoti grūti palielināt sakarā ar to, ka eksistē ieraksti, kuriem adrese nav noradīta vai arī TAB_ADDRESS tabulā neeksistē ierakstu, kur adrese ir līdzīga uzņēmuma adresei. Piemēram, uzņēmuma adresi ‘Lidosta “Rīga”’ nevar piesaistīt nevienam ierakstam adrešu tabulā. Var sanākt problēmas, ja, piemēram mājas numurs kaut kāda iemesla dēļ vai nav noradīts vai ir nepareizs, bet vissvarīgākais iemesls ir TAB_ADDRESS nepilnība.

TC.address	TC.x	TC.y	TC.addrid	TA.name	TA.namenum
K. Barona 130	545257	603575	45600	Krišjāņa Barona iela	130
Tērbatas 34 - 1a	541438	606509	53129	Tērbatas iela	34
Aspazijas bulvāris 5	540018	608542	52581	Aspazijas bulvāris	5
Lāčplēša 18	541173	606431	54283	Lāčplēša iela	18
A. Deglava 60	549090	607786	37064	Augusta Deglava iela	60
Miera 10	542292	604808	54032	Miera iela	10
Elizabetes 41/43	539588	605890	54294	Elizabetes iela	41/43
Ģertrūdes 94a	543521	608403	33346	Ģertrūdes iela	94A
Viktorijas 33, p/n Majori	494003	601807	2220	Viktorijas iela	33
K. Valdemāra 38 - 202. kab.	540702	605261	55110	Krišjāņa Valdemāra iela	38
Matīsa 125	545393	608162	55072	Matīsa iela	125
Mazā Nometņu 34a	533144	611968	44136	Mazā Nometņu iela	34A
Tīnūžu 1	553820	612029	40275	Tīnūžu iela	1
Vidusskola, p/n Madliena	null	null	null	-	-
Tvaika 52	null	null	null	-	-
Marijas ielas tunelis	null	null	null	-	-

Tabula 2. TAB_CLIENTS un TAB_ADDRESS saite

Tabulas 2. pirmās četras kolonas attēlo TAB_CLIENTS (TC) tabulas fragmentu ar aizpildītiem *addrid*, *x* un *y* laukiem, bet pēdējie ir TAB_ADDRESS (TA) tabulas *name* un *namenum* lauku saturi, pie tam adrešu tabulas ierakstu identifikatori sakrīt ar uzņēmumu tabulas *addrid* lauku saturu. Var redzēt ka tabulu saite ir korekta.

4.2.3. Izmaiņas atjaunošanas procesā

Datu atjaunošana notiek divreiz nedēļā. Adreses maiņa notiek apmēram 100-200 uzņēmumiem katra atjaunošanas reizē. Izpildīt *addressupdate* visiem ierakstiem katru nedēļu nav iespējams, jo procedūras izpildes laiks ir lielāks par visa atjaunošanas procesa laiku, tāpēc ļoti svarīgi ir pievērst lielo uzmanību uzņēmumiem, kuriem atjaunošanas laikā mainās adrese.

Uzņēmumu tabulas atjaunošana notiek ar *ltkcheckclient* procedūras palīdzību (sk. nodaļā 4.1.1). Sakarā ar tabulu izmaiņām, procedūrai jāpievieno sekojošas darbības: pirmkārt, jāpārbauda vai vecā adrese atšķiras no jaunā, vai abi tie ir vienādi. Ja viņi tomēr nesakrīt, jāpārveido *specaddr* lauku, kā arī jāizpilda gandrīz tādas pašas darbības kā *addressupdate* procedūrā (sk. nodaļā 4.2.2.1). Ja jaunas adreses pirmais vārds nav viens no 'Mazā', 'Jaunā' un tā tālāk, tad procedūra meklē ierakstu adrešu tabulā, kur *easyname* lauks ir nepieciešama TAB_CLIENTS tabulas ieraksta *specaddr* lauka daļa, tas nozīmē, ka kreisā robeža nav stingra. Adreses meklēšana šajā gadījumā notiek sekojoši:

```
select top 1 id_address,x,y into idspec,xspec,yspec
  from tab_address
  where locate(specc,easyname) > 0 and city = speccity;
```

Kur *idspec*, *xspec*, *yspec* ir numeric tipa mainīgie, *specc*- uzņēmumu tabulas ieraksta jaunā adreses speciālā forma (*specaddr* lauks vērtība), *speccity*- uzņēmumu tabulas ieraksta pilsēta. Pieprasījums atgriež tieši viena TAB_ADDRESS tabulas ieraksta identifikatoru un koordinātes, pie tam jāpārbauda lai pilsētas lauks (*city*) arī būtu vienāds ar uzņēmumu tabulas ieraksta pilsētu.

Ja tomēr ieraksta pirmais vārds ir 'slikts', pieprasījums mazliet atšķiras no iepriekšēja:

```
select top 1 id_address,x,y into idspec,xspec,yspec
  from tab_address
  where locate(specc,easyname) = 1 and city = speccity
```

Var redzēt ka atšķirībā no pirmā pieprasījuma *locate* (funkcija kas atgriež rindas pozīciju citā rindā) vērtība jābūt vienāda ar 1. Tas nozīme, ka adreses speciālai formai jāsākas ar *easyname* lauka saturu. Tā piemēram, ja *specc* ir vienāds ar '.maza.nometnu.1.-.17.', tad pieprasījumam jāatgriež adrešu tabulas ieraksta identifikatoru un koordinātes, kur *easyname* lauks ir '.maza.nometnu.1.'.

Kad TAB_ADDRESS tabulas ieraksta identifikators un koordinātes ir atrasti paliek tikai aizpildīt TAB_CLIENTS tabulas ieraksta atbilstošus laukus.

```
update admin.tab_clients set addrid = idspec,x = xspec,y = yspec
  where ltk_code = ltk_cd
```

Kur *ltk_code* ir ieraksta identifikators Lattelekom datu bāzē, kas saista TAB_LTKIMPORT, TAB_LTKFULLBASE, TAB_LTKCLIENTS un TAB_CLIENTS tabulas,

bet *ltk_cd* ir procedūras parametrs, kas nozīmē tekoša ieraksta identifikatoru.

Pēc aprakstītām izmaiņām *ltkcheckclient* procedūrā, ja gadījumā atjaunošanas laikā tiek mainīta uzņēmuma adrese, *addrid*, *x*, *y* lauku atjaunošana notiek automātiski. Tātad nav nepieciešami izsaukt kaut kādus papildus procedūras. Līdz ar to tabulu pārveidošana ir pabeigta.

4.3. Maršrutu attēlojums kartē

Šīs jaunās iespējas realizācijas sastāv no vairākiem soļiem. Pirmais- izdomāt visērtāko tabulu struktūru, lai būtu iespējams redzēt ne tikai pašus maršrutus, bet arī pieturu un laiku sarakstus atbilstošiem pilsētas un piepilsētas maršrutiem; otrais- tabulu aizpildīšana, kas šajā gadījumā ietver datu kopēšanu no Rīgas Satiksmes datu bāzes un to pārveidošanu vajadzīgā formātā; un pēdējais- papildus iespēju aplūkošana un daļēja realizācija, kā piemēram dažādu maršrutu atrašana, kas savieno divus punktus Rīgas robežās.

Lai saprastu kāda datu bāzes struktūra ir nepieciešama, vajag apskatīt kā gatavs projekts izskatīsies dzīvē. Portālā būs pieejami vairāki saraksti: trolejbusu, autobusu un tramvaju maršruti, pie tam viņiem var būt piesaistīti arī papildus maršruti, kā piemēram 16. autobusam ir pieejami divi papildus virzieni: Abrenes iela – Garkalnes Mucenieki un Mucenieki – Katedrāle, bet 5. tramvajam četri: 3.depo – Iļģuciems, 3.depo – Mīlgrāvis – Iļģuciems, 4.depo – Iļģuciems un 4.depo – Mīlgrāvis – Iļģuciems. Izvēlot konkrēto maršrutu vai papildus maršrutu kartē paradīsies līnija, kas attēlo maršrutu un tabula ar atbilstoša maršruta pieturām. Pie tam vajag ņemt vērā, ka maršrutam ir divi virzieni un to attēlojums interaktīvajā kartē atšķiras, tātad līnija sastāvēs no divām daļām, tas nozīme: ja maršruta galapunktus apzīmēt ar X un Y burtiem, tad pirmā daļa būs X→Y, otrā Y→X. Tas pats attiecās uz tabulu ar pieturām. Ir vajadzīgi divās kolonas, kas atbilst diviem virzieniem.

Lietotājs var izvēlēt pieturu no saraksta un apskatīt transporta līdzekļa atiešanas laikus no šīs pieturas. Ja atiešanas laiki darba dienās un brīvdienās atšķiras, tad var pārslēgt maršruta režīmus.

Līdz ar dotu aprakstu ir skaidrs, ka pirmā etapa realizācijai ir nepieciešamas trīs tabulas: maršrutu tabula, punktu un pieturu tabula, laiku tabula.

4.3.1. Datu struktūras ieviešana

Galvenā no jaunizveidotajām tabulām ir maršrutu tabula (sk. Zīmējumu 8.). Tas nosaukums ir TAB_MARSHRUTS un tā saturēs informāciju par maršrutiem, to režīmiem, virzieniem un citu informāciju.

Name	Type	Column ID	Allows Nulls
id_marshruts	numeric(9,0)	1	No
number	char(50)	2	Yes
name	char(100)	3	Yes
attribute	char(20)	4	Yes
sequence	integer	5	Yes
marshtype	integer	6	Yes
pretname	char(100)	7	Yes
transports	integer	8	Yes
mainmarsh	integer	9	Yes

Zīmējums 8. TAB_MARSHRUTS

Tabula satur sekojošas kolonas:

1. *id_marshruts* – tabulas primārā atslēga, unikāls identifikators
2. *number* – maršruta numurs
3. *name* – maršruta nosaukums, jeb virziens
4. *attribute* – lauks, kas nosaka ierakstam atbilstoša maršruta kursēšanas dienas (1-5 – darba dienas, 6-7 – brīvdienas, 6 – sestdiena, 7 – svētdiena, 1-7 – visas dienas)
5. *sequence* – maršruta numurs skaitliskā formā, piemēram, 3, 3a un 3e autobusiem *sequence*= 3
6. *marshtype* – maršruta tips, kas nosaka laika saraksta režīmu: 1 – katru dienu, 2 – darba dienas/brīvdienas, 3 – darba dienas/sestdiena/svētdiena, 4 – darba dienas/sestdienas, 5 – darba dienas, 6 – brīvdienas
7. *pretname* – maršruta otra virziena nosaukums
8. *transports* – transporta veids: 1 – autobuss, 2 – reģionāls autobuss, 3 – tramvajs, 4 - trolejbuss
9. *mainmarsh* – galvenā maršruta identifikators; aizpildīts tikai papildus maršrutiem

Atbilstoši tabulas struktūrai: ja kādam maršrutam ir dažādi laika saraksti darba dienās, sestdienās un svētdienās, tad TAB_MARSHRUTS tabulā tas atkārtosies trīs reizēs ar dažādiem *attribute* lauka vērtībām (1-5, 6 un 7), bet *marshtype* visiem trijiem būs vienāds ar 3.

Otra jauna tabula ir - TAB_MARSHCOORD (sk. Zīmējumu 9.), kas ir nepieciešama maršrutu punktu un pieturu informācijas glābšanai. Papildus kartes punkti ir nepieciešami lai, savienojot tos kopā ar pieturām, iegūt maršruta līnijas. Atbilstoši šīs tabulas datiem tiks veidots arī pieturu saraksts.

Name	Type	Column ID	Allows Nulls
id_marshcoord	numeric(9,0)	1	No
marshruts	numeric(9,0)	2	Yes
x	float	3	Yes
y	float	4	Yes
lvl	integer	5	Yes
name	char(100)	6	Yes
type	integer	7	Yes
equal	integer	8	Yes
place	integer	9	Yes
direction1	integer	10	Yes
direction2	integer	11	Yes

Zīmējums 9. TAB_MARSHCOORD

TAB_MARSHCOORD kolonas:

1. *id_marshcoord* – tabulas primārā atslēga
2. *marshruts* – ārēja atslēga no TAB_MARSHRUTS tabulas, maršruta identifikators
3. *x* – punkta x koordināte
4. *y* – punkta y koordināte

5. *lvl* – līmenis, no kura punkts sāk rādīties kartē (kartēm ir 6 līmeņi, ja ieraksta *lvl*=0, tad punkts rādās visos līmeņos, ja *lvl*=5 tad tikai visprecīzākā līmenī); lauks ir nepieciešams datu ātrākai atlasei
6. *name* – punkta nosaukums; ir aizpildīts tikai pieturām
7. *type* – tips: pieturām ir vienāds ar 4, pārējiem nav aizpildīts (null)
8. *equal* – pieturas identifikators, kas ļauj atrast kopīgas pieturas dažādiem maršrutiem, piemēram, visos maršrutos Abrenes ielas *equal* lauks būs viens un tāds pats
9. *place* – punkta vieta maršrutā; lauks ir nepieciešams lai atrast pareizo punktu secību
10. *direction1* – punkta virziens, kas var būt vienāds vai nu ar 1, vai 2
11. *direction2* – lauks, kas ļauj atrast visus galapunktus: galapunktiem ir vienāds ar 1, pārējiem punktiem un pieturām ar 0

Bez šaubām TAB_MARSHCOORD tabula būs daudzreiz lielāka nekā aprakstīta TAB_MARSHRUTS tabula, jo katram maršrutam ir nepieciešami vairāki punkti, kā arī pieturas lai izveidotu līniju kartē un pieturu sarakstus abos virzienos.

Pēdēja jauna tabula ir TAB_MARSHTIME tabula (sk. Zīmējumu 10.), kas saturēs laika sarakstus jeb atiešanas laikus no katras pieturas.

Name	Type	Column ID	Allows Nulls
id_marshtime	integer	1	No
marshcoord	numeric(9,0)	2	Yes
h5	char(600)	3	Yes
h6	char(600)	4	Yes
h7	char(600)	5	Yes
h8	char(600)	6	Yes
h9	char(600)	7	Yes
h10	char(600)	8	Yes
h11	char(600)	9	Yes
h12	char(600)	10	Yes
h13	char(600)	11	Yes
h14	char(600)	12	Yes
h15	char(600)	13	Yes
h16	char(600)	14	Yes
h17	char(600)	15	Yes
h18	char(600)	16	Yes
h19	char(600)	17	Yes
h20	char(600)	18	Yes
h21	char(600)	19	Yes
h22	char(600)	20	Yes
h23	char(600)	21	Yes
h0	char(600)	22	Yes
h1	char(600)	23	Yes
h2	char(600)	24	Yes

Zīmējums 10. TAB_MARSHTIME

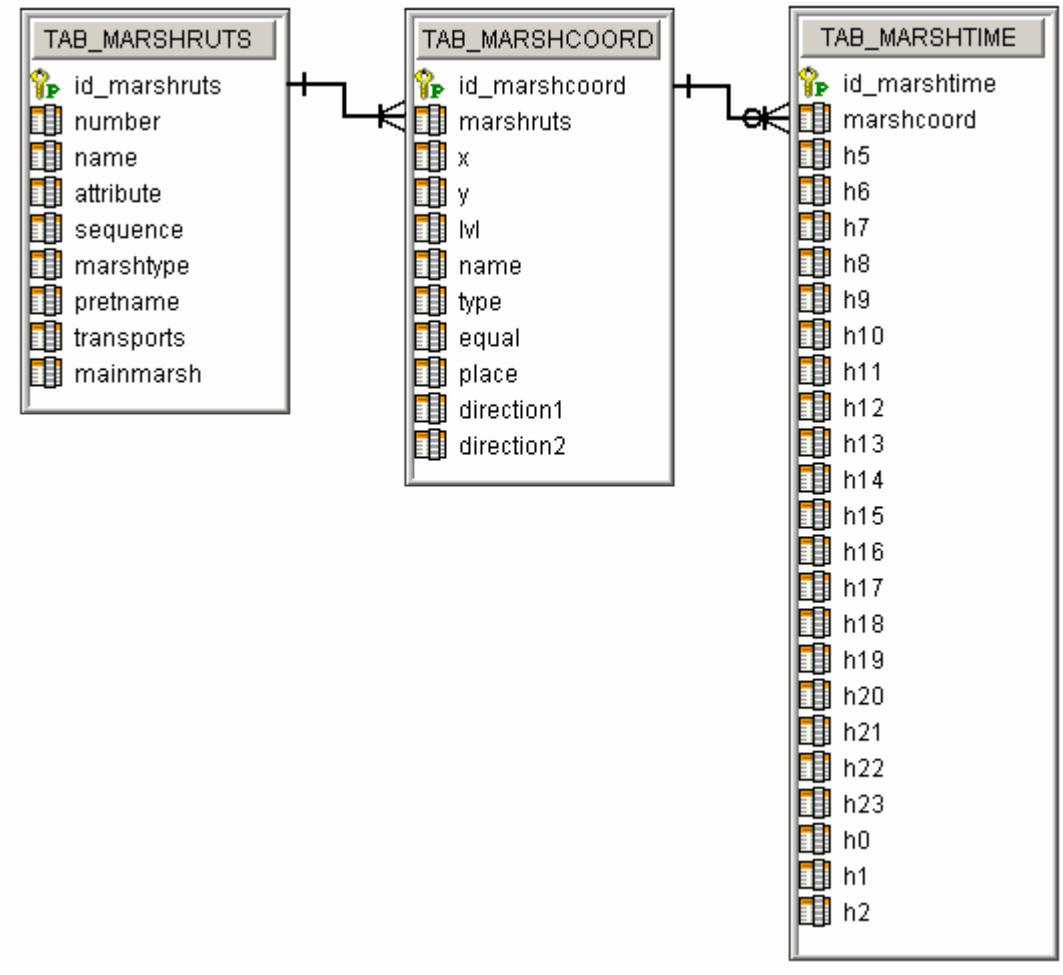
TAB_MARSHTIME satur:

1. *id_marshtime* – tabulas primārā atslēga
2. *marshcoord* – ārēja atslēga no TAB_MARSHCOORD tabulas, punkta identifikators

3. – 24. ***hn*** – laika saraksts; minūtes, kuros transports atiet no dota pietura pulksten ***n*** (***n*** – naturāls skaitlis mazāks vai vienāds ar 23), formātā 'min1 min2 min3 min4...'. Piemēram, *h5*='16 25 39' nozīmē, ka autobusa, trolejbusa vai tramvaja atiešanas laiki no sekojoša punkta ir: 5:16, 5:25, 5:39

Var redzēt kā lauku ar nosaukumiem h3 un h4 TAB_MARSHTIME tabulā neeksistē. Tas ir saistīts ar to, ka šie lauki nav nepieciešami. Saskaņā ar Rīgas Satiksmes datu bāzes datiem nav maršrutu, kur atiešanas vai izbraukšanas laiks būtu trijos vai četros naktī. Laiks TAB_MARSHTIME tabulā eksistēs tikai punktiem, kas ir pieturas. No šīs tabulas datiem galējā produktā veidosies tabula ar maršrutu atiešanas laikiem no pieturām.

Saites starp jaunajām TAB_MARSHRUTS, TAB_MARHSCOORD un TAB_MARSHTIME tabulām var apskatīt Zīmējumā 11.



Zīmējums 11. Jaunais ER modelis

Var secināt, ka katram maršrutam no TAB_MARSHRUTS tabulas eksistē vismaz viens punkts TAB_MARSHCOORD tabulā, bet tikai dažiem punktiem (reālajā dzīvē – pieturām) eksistē rindas laiku tabulā.

4.3.2. Maršrutu tabulu aizpildīšana

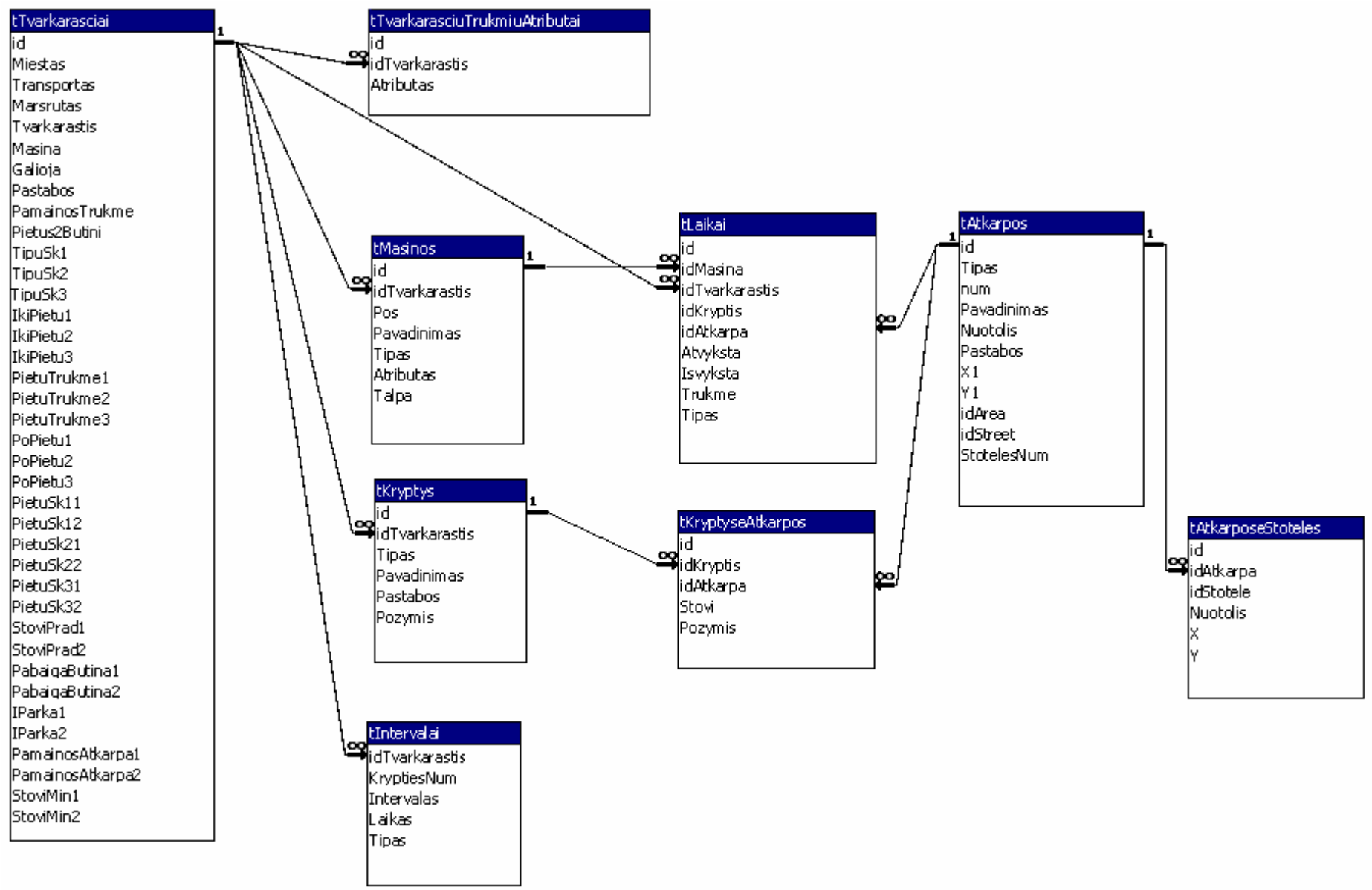
TAB_MARSHRUTS, TAB_MARSHCOORD un TAB_MARSHTIME tabulu aizpildīšana

notiks saskaņā ar Rīgas Satiksmes datu bāzes datiem. Lai saprastu uzdevumus jāapskata tās datu bāzi un tabulu struktūru.

4.3.2.1. Rīgas Satiksmes datu bāzes apraksts

SIA Rīgas Satiksme ir Rīgas pašvaldības uzņēmums, kas dibināts 2003. gada 20. februārī. Jau vairākus gadus tas piedāvā savu lappusi Internētā, kur var apskatīt gan pilsētas, gan piepilsētas autobusu maršrutu sarakstu, kā arī to laika sarakstus. 2005. gada janvārī apvienojās četri Rīgas pašvaldības sabiedriskā transporta uzņēmumi - SIA Rīgas Satiksme, SIA Tramvaju un trolejbusu pārvalde, SIA Rīgas autostāvvietas un SIA Rīgas domes autobāze. Tādējādi RP SIA Rīgas Satiksme kļuva par lielāko pilsētas pašvaldības uzņēmumu. Līdz ar apvienošanas Rīgas Satiksmes portālā Internetā kļuva pieejami arī trolejbusu un tramvaju maršruti . Visi sistēmas dati tiek ņemti no Rīgas Satiksmes datu bāzes [14].

Rīgas Satiksmes datu bāze tiek realizētā Access vidē un satur 9 tabulas, kur glabājas informācija par visiem Rīgas un piepilsētas maršrutiem, to punktiem un pieturām, ka arī transportlīdzekļu atiešanas laikiem. Tās būs nepieciešamas lai aizpildītu vajadzīgas tabulas 1188 datu bāzē. Zīmējumā 12 ir parādītās šīs tabulas, kā arī relācijas starp tām.



Zīmējums 12. Rīgas Satiksmes datu bāze

Tālāk ir dots Rīgas Satiksmes (RS) datu bāzes tabulas un to lauku apraksts. Katrai apraksta tabulai ir sekojošas kolonas: RS datu bāzes tabulas kolonas nosaukums, nosaukums angļu valodā, kolonas tips, kolonas apraksts.

TTvarkarasciai (sk. Tabulu 3.) – maršrutu tabula.

tTvarkarasciai	**tShedules**		
id	**id**	long	skaitītājs, kas automātiski palielinās, kad tabulai tiek pievienota jaunā rinda; tTvarkarasciai tabulas primārā atslēga
Miestas	**City**	char(20)	pilsētas nosaukums
Transportas	**Division**	char(20)	transportlīdzekļa nosaukums: Autobuss, Autob_reg, tramvajs vai trolejbuss
Marsrutas	**Line**	char(50)	maršruta numurs
Tvarkarastis	**Version**	char(20)	maršruta tips, kas ir nepieciešams lai atšķirt vairākus maršrutus ar vienādu numuru, bet dažādiem laika sarakstiem, piemēram, darba dienās un brīvdienās
Galioja	**Date**	date	lietotāja noradītais datums (pievienošanas datums)
Masina	**Vehicle**	char(50)	transportlīdzekļu atslēgu vārdu saraksts, formātā: tr1,tr2,tr3; piemēram: Autob,Expres,...
Pastabos	**Comment**	char(255)	lietotāja komentāri

Tabula 3. tTvarkarasciai

TTvarkarasciai tabulas lauki, kas nav aprakstīti 3.tabulā ir nepieciešami tikai Rīgas Satiksmes speciālās programmas izmantošanai.

tIntervalai (sk. Tabulu 4.) – laika intervālu tabula.

tIntervalai	**tHeadways**		
idTvarkarastis	**idShedule**	long	arēja atslēga no tTvarkarasciai tabulas - maršruta identifikators
KryptiesNum	**PatternNum**	int	virziena numurs; 1 – nozīmē A>B virzienu, 2 - B>A virzienu
Intervalas	**Headway**	single	nepieciešams vidējais intervāls
Laikas	**Daytime**	long	sākuma laiks vidējam intervālam
Tipas	**VehiclesCount**	int	maršruta transportlīdzekļu skaits; ir nepieciešams RS speciālās programmas vajadzībām

Tabula 4. tIntervalai

tMasinos (sk. Tabulu 5.) – transportlīdzekļu tabula, kas satur informāciju par konkrētiem autobusiem, trolejbusiem un tramvajiem.

tMasinos	**tVehicles**		
id	**id**	long	skaitītājs, kas automātiski palielinās, kad tabulai tiek pievienota jaunā rinda; tMasinos tabulas primārā atslēga
idTvarkarastis	**idShedule**	long	arēja atslēga no tTvarkarasciai tabulas - maršruta identifikators
Pos	**Pos**	integer	izbraukšanas pozīcija maršrutā
Pavadinimas	**Name**	char(20)	transportlīdzekļa nosaukums vai numurs
Tipas	**Type**	long	transportlīdzekļa kursēšanas tips: 0 –agrās un vēlās braukšanas, 1 –tikai agrās braukšanas, 2 –tikai vēlās braukšanas, 3 –jauktās braukšanas
Atributas	**Keyword**	char(50)	transportlīdzekļa atslēgas vārds
Talpa	**Capacity**	single	Transportlīdzekļa ietilpība

Tabula 5. tMasinos

tLaikai (sk. Tabulu 6.) – vissvarīgākā un vislielākā tabula, kas kontrolē visu maršrutu katras mašīnas pārvietošanu; laiku tabula.

tLaikai	**tRuns**		
id	**id**	long	skaitītājs, kas automātiski palielinās, kad tabulai tiek pievienota jaunā rinda; tLaikai tabulas primārā atslēga
idMasina	**idVehicle**	long	arēja atslēga no tMasinos tabulas - transportlīdzekļa identifikators
idTvarkarastis	**idShedule**	long	arēja atslēga no tTvarkarasciai tabulas - maršruta identifikators
idKryptis	**idPattern**	long	arēja atslēga no tKryptys tabulas - virziena identifikators
idAtkarpa	**idLink**	long	arēja atslēga no tAtkarpa tabulas - punkta identifikators
Atvyksta	**Arrives**	single	ierašanas laiks
Isvyksta	**Departs**	single	izbraukšanas laiks
Trukme	**Duration**	single	braukšanas laiks līdz posma beigām, jeb posma izbraukšanas ilgums minūtēs
Tipas	**Tag**	long	transporta konkrēta brauciena darba režīms: 0 - autobuss maršrutā, 1 - nulles reisā piešķirtais posma laiks = 0, 2 - šoferu maiņas brīdis, 4 - pusdienas laiks 1. maiņai, 8 - autobuss maršrutā, 9 - nulles reiss, 12 - pusdienas laiks 2. maiņai, 16 – pārtraukums; nepieciešamie izmantojamie tipi parastiem maršrutiem ir 0 un 8

Tabula 6. tLaikai

tKryptys (sk. Tabulu 7.) – tabula, kas satur informāciju par maršrutu virzieniem, pie tam katram maršrutam ir vismaz divi virzieni: no viena galapunkta uz otro un pretēja virzienā.

tKryptys	**tPatterns**		
id	**id**	long	skaitītājs, kas automātiski palielinās, kad tabulai tiek pievienota jaunā rinda; tKryptys tabulas primārā atslēga
idTvarkarastis	**idShedule**	long	arēja atslēga no tTvarkarasciai tabulas - maršruta identifikators
Tipas	**Type**	char(5)	virziena tips - A>B, B>A, D>A, D>B, A>D, B>D, A1>B2...
Pavadinimas	**Name**	char(50)	virziena nosaukums
Pastabos	**Comment**	char(255)	lietotāja komentāri
Pozymis	**Tag**	char(20)	norādītājs, tukšiem braucieniem Pozymis=0

Tabula 7. tKryptys

Parasti vienkāršie braucieni no viena galapunkta līdz otram un otrādi ir ierakstīti tKryptys tabulā ar *Tipas*=’A>B’ un *Tipas*=’B>A’, braucieni no parka vai depo un pretējā virzienā ar *Tipas*=’A>D’ un ‘D>A’, bet pārējie braucieni ar mainītiem galapunktiem ar *Tipas*=’A**n**>B**m**’, kur **n** un **m** naturālie skaitļi.

tKryptyseAtkarpos (sk. Tabulu 8.) – tabula kas savieno virzienu un punktu tabulas, aizpildīta tikai dažiem punktiem.

tKryptyseAtkarpos	**tPatternLinks**		
id	**id**	long	skaitītājs, kas automātiski palielinās, kad tabulai tiek pievienota jaunā rinda; tKryptyseAtkarpos tabulas primārā atslēga
idKryptis	**idPattern**	long	arēja atslēga no tKryptys tabulas - virziena identifikators
idAtkarpa	**idLink**	long	arēja atslēga no tAtkarpa tabulas - punkta identifikators
Stovi	**Pause**	long	nepieciešama pauze pirms braukšanas uz posmu
Pozymis	**Tag**	char(20)	norādītājs, tukšiem braucieniem Pozymis=0

Tabula 8. tKryptyseAtkarpos

Atkarpos (sk. Tabulu 9.) – tabula, kas satur informāciju par visiem punktiem.

tAtkarpos	**tLinks**		
id	**id**	long	skaitītājs, kas automātiski palielinās, kad tabulai tiek pievienota jaunā rinda; tAtkarpos tabulas primārā atslēga
Tipas	**Type**	long	objekta tips 1-nogrieznis,2-iela,3-rajons,4-pietura,5-krustojums,6-cita tipa punkts
num	**Num**	char(10)	numurs vai nosaukums lietotāja papildus vajadzībām
Pavadinimas	**Name**	char(50)	objekta nosaukums
Pastabos	**Comment**	char(50)	lietotāja komentāri
X1	**X1**	single	X koordināte kartē
Y1	**Y1**	single	Y koordināte kartē

Tabula 9. tAtkarpos

tAtkarposeStoteles (sk. Tabulu 10.) – tabula, kas satur informāciju par maršrutu braukšanas posmiem.

tAtkarposeStoteles	**tLinkSets**		
id	**id**	long	skaitītājs, kas automātiski palielinās, kad tabulai tiek pievienota jaunā rinda; tAtkarposeStoteles tabulas primārā atslēga
idAtkarpa	**idOwner**	long	arēja atslēga no tAtkarpa tabulas – galvenā punkta identifikators
idStotele	**idMember**	long	papildus punkta identifikators
Nuotolis	**Distance**	single	attālums no iepriekšēja punkta
X	**X**	single	X koordināte kartē
Y	**Y**	single	Y koordināte kartē

Tabula 10. tAtkarposeStoteles

tTvarkarasciuTrukmiuAtributai (sk. Tabulu 11.) – maršrutu atribūtu tabula.

TTvarkarasciu-TrukmiuAtributai	**TShedules-Durations-Keywords**		
id	**id**	long	skaitītājs, kas automātiski palielinās, kad tabulai tiek pievienota jaunā rinda; tTvarkarasciuTrukmiuAtributai tabulas primārā atslēga
idTvarkarastis	**dShedule**	long	arēja atslēga no tTvarkarasciai tabulas - maršruta identifikators
Atributas	**Keywords**	char(255)	atribūts, kas attēlo maršruta kursēšanas dienas: 1-5 – darba dienas, 6-7 – brīvdienas, 6 – sestdiena, 7 – svētdiena, 1-7 – visas dienas

Tabula 11. tTvarkarasciuTrukmiuAtributai

Apskatīsim vienu piemēru – 19 trolejbusa maršrutu. Lai atrast tā informāciju jāsāk meklēt

šo maršrutu tTvarkarasciai tabulā.

TTvarkarasciai tabulā 19 trolejbuss atkārtojas divas reizes:

id	miestas	transportas	marsrutas	tvarkarastis	masina	galioja
1059	Riga	trolejbuss	19	1	trolejbuss	08.09.2005
1060	Riga	trolejbuss	19	5	trolejbuss	02.05.1998

Lai atrast kurš no maršrutiem atbilst darba dienām, bet kurš brīvdienām vajag atrast atbilstošas rindas tTvarkarasciuTrukmiuAtributai, kur *idTvarkarastis* sakrīt ar iepriekšējo rezultātu, tātad ir vienāds vai nu ar 1059, vai 1060.

Id	idtvarkarastis	atributas
1348	1059	1-5
1365	1060	6-7

1-5 nozīmē, ka maršruts kursē no pirmdienas līdz piektdienai, bet 6-7 – no sestdienas līdz svētdienai. Tātad var secināt, ka maršruts ar identifikatoru 1059 atbilst 19 trolejbusa darba dienu kursēšanas režīmam, bet 1069- brīvdienu režīmam. Tā kā punkti un pieturas vienam maršrutam dažādas dienās neatšķiras, tālāk skatīsimies tikai darba dienu maršrutu. Nākošais etaps- atrast maršruta virzienus no tKryptys tabulas, kur maršruts sakrīt ar 19 maršrutu darba dienās, tātad *idtvarkarastis*=1059.

id	idtvarkarastis	tipas	pavadinimas
7295	1059	A>B	Pētersalas iela - Ziepniekkalns
7296	1059	B>A	Ziepniekkalns - Pētersalas iela
7297	1059	A>D	Pētersalas iela - D
7298	1059	D>B>A	D - Ziepniekkalns-Pētersalas iela
7299	1059	D>A	D - Pētersalas iela
7300	1059	A>B>D	Pētersalas -Ziepniekkalns - D
7301	1059	D>B>A	D1- Ziepniekkalns-Pētersalas iela
7302	1059	A>B>D	Pētersalas -Ziepniekkalns - D1

Galvenie virzieni ir pirmais un otrais (ar tipiem A>B un B>A), pārējie atbilst maršrutiem uz parkiem no Pētersalas ielas (A>...) un no parkiem uz Pētersalas ielu (D>..., D1>...). D un D1 ir divi dažādi trolejbusu parki.

Lai atrastu maršrutu no Pētersalas ielas uz Ziepniekkalnu no tLaikai vajag atlasīt tikai rindas kur *idTvarkarastis*=1059, *idKryptis*=7295 un *Tipas* ir vai nu 0, vai 8. Pie tam vajag paņemt kādu konkrētu *idMasina* vērtību, jo pretējā gadījumā viņš atradīs visu transportlīdzekļu kursēšanas laikus. Paņemsim *idMasina*=4844. Rezultātu sakārto pēc ierašanas laika- *Atvyksta*.

id	idmasina	idtvarkarastis	idkryptis	idatkarpa	atvyksta	isvyksta	trukme
4077853	4844	1059	7295	14810	330	330	0.1
4077854	4844	1059	7295	9804	330.1	330.1	1.3
4077855	4844	1059	7295	9798	331.4	331.4	1.5
4077856	4844	1059	7295	9792	332.9	332.9	1.4
4077857	4844	1059	7295	9729	334.3	334.3	0.7
4077858	4844	1059	7295	9731	335	335	1.9

4077859	4844	1059	7295	10000	336.9	336.9	0.2
4077860	4844	1059	7295	10001	337.1	337.1	0.8
4077861	4844	1059	7295	10002	337.9	337.9	0.6
4077862	4844	1059	7295	10004	338.5	338.5	1

...

Atvyksta un *Isvyksta* ir laiks minūtēs no diennakts sākuma. Katrs maršruta virziens sastāv no daudziem posmiem (tAtkarposStoteles tabula), savukārt posmi sastāv no pieturvietām, krustojumiem, ielu nosaukumiem, rajona nosaukuma (tAtkarpos tabula). Tabulā tLaikai laukā *Atvyksta* glabājās laiks, cikos autobuss iebrauc konkrētajā posmā un laukā Trukme posma izbraukšanas ilgums minūtēs. Zinot ilgumu un pieturvietas attālumu no posma sākuma, var precīzāk izrēķināt katras pieturvietas laiku.

Lai atrast maršruta punktus un pieturas vajag atlasīt no tAtkarposStoteles tabulas posmu galvenās jeb sākumu punktus (tLaikai.idAtkarpa=tAtkarposStoteles.idAtkarpa), kā arī pieturas no tAtkarpos tabulas, ja posma beigu punkts ir pietura (tAtkarposStoteles.idstotele = tAtkarpos.id and tAtkarpos.tipas = 4). Tā var atrast visu punktu koordinātes un pieturu nosaukums, kā arī pareizo punktu kārtību. X un y koordinātes Rīgas Satiksmes datu bāzē atšķiras no karšu koordinātēm 1188 datu bāzē, bet viņus var izrēķināt ar formulu palīdzību: 1188.x=RS.x*1000*2.5-727500, 1188.y=RS.y*1000*(-2.5)+16387500, kur 1188.x un 1188.y ir koordinātes 1188 datu bāzē, bet RS.x un RS.y- Rīgas Satiksmes datu bāzē.

Par maršruta otro virzienu ir nepieciešams ņemt vērā, ka galapunktā, braucot atpakaļ, mainās braukšanas virziens tabulā tLaikai lauks *idKryptis*. Galapunktā transportlīdzeklis vienmēr mazliet pastāv. Laukā *Atvyksta* ir ierašanās laiks iepriekšējam virzienam, bet laukā *Isvyksta* izbraukšanas laiks jaunajam virzienam.

4.3.2.2. Datu kopēšana no Rīgas Satiksmes datu bāzes

Runājot par datu kopēšanu un pārveidošanu nepieciešamā formātā, pirmais uzdevums ir visu tabulu kopēšana no Rīgas Satiksmes datu bāzes 1188 datu bāzē. To var izdarīt Accessā, noeksportējot katru tabulu atsevišķi ODBC formātā uz nepieciešamo Sybase datu bāzi. Pēc šīs procedūras visas primārās un ārējās atslēgas pazūd, līdz ar to Sybase tos, kā arī tabulu indeksus jāuztaisa no jaunas. Nākamais solis ir pats apjomīgs. Tas iekļauj TAB_MARSHRUTS, TAB_MARSHCOORD un TAB_MARSHTIME tabulu aizpildīšanu, kas ietver Rīgas Satiksmes datu pārveidošanu. Šim mērķim ir uzrakstīta speciālā procedūra- *getmarsh*. Pie tam procedūras ideja ir līdzīga iepriekšējam piemēram- 19.trolejbusa apskatīšanai.

Getmarsh procedūras uzdevumi ir atlasīt visus maršrutus, atlasīt visus punktus un pēdējais- atlasīt pieturas laikus. Šiem mērķiem tika izmantoti kursori.

Pirmais kursors ir nepieciešams lai atrastu visus maršrutus. Viņš atlasa no Rīgas Satiksmes tabulām 1188 datu bāzes maršruta identifikatoru, transportlīdzekļa veidu, maršruta virzienu (tā nosaukumu un identifikatoru), maršruta tipu, maršruta atribūtu, jeb kursēšanas dienas. Šie dati būs nepieciešami TAB_MARSHRUTS tabula aizpildīšanai. Šinī tabulā transportlīdzekļa veids kodējas ar skaitļi: 1, 2, 3 vai 4, atbilstoši: autobuss, reģionāls autobuss, tramvajs, trolejbuss, tāpēc ērtāk būs uzreiz kodēt *transportas* lauka vērtību uz skaitļi.

Lai atrastu visus papildus maršrutus, kur virziens nav vienāds ar ‘A>B’ un ‘B>A’ tika izveidota speciālā tabula TAB_TEMPMARSH ar trīs laukiem: *idmarsh*, *idkr*, *idpretkr*, kur glabāsies informācija par papildus maršrutu identifikatoriem Rīgas Satiksmes datu bāzē (tTvarkarasciai.id), maršruta virziena identifikators (tKryptys.id) un pretēja virziena identifikators, ja tas eksistē. Šī tabula ir nepieciešama lai pievienot TAB_MARSHRUTS tabulai arī šos maršrutus. Uz šo tabulu tiek kopēti visi maršruti, kam eksistē papildus virzieni (t.Kryptys.Tipas in (‘A>B1’, ‘A>B2’, ‘A>C’, ‘A>D’, ‘B>D’, ...)). B**n** un A**n**, kur **n** – naturāls skaitlis**,** virziena tipā nozīmē ka

maršrutam ir cits galapunkts, bet D – parks vai depo. Ja papildus maršrutam ir abi virzieni, piemēram, ‘A1>B1’ un ‘B1>A1’, tad TAB_TEMPMARSH tabulā visi rindas lauki būs aizpildīti ar datiem, pretējā gadījumā *idpretkr*=null. Tika pieņemts, ka visiem maršrutiem, kuri brauc uz parku vai depo vai otrādi, ir tikai viens virziens - *idkr*.

Tātad kursors *getmarsh* procedūrā izskatās sekojoši:

```
declare vw_cur dynamic scroll cursor for
  select t1.id as tid, (case transportas
                   when 'Autobuss' then '1'
                   when 'Autob_reg' then '2'
                   when 'Tramvajs' then '3'
                   when 'Trolejbuss' then '4' end) as Transport,
         t1.marsrutas, t2.pavadinimas, t2.id as kid, t1.tvarkarastis,
         t3.atributas
  from admin.tTvarkarasciai as t1, admin.tkryptys as t2,
       admin.tTvarkarasciutrukmiuatributai as t3
  where t2.tipas = 'A>B' and t1.id = t2.idtvarkarastis and
        t3.idTvarkarastis =  t1.id
  union
  select idmarsh as tid, (case transportas
                   when 'Autobuss' then '1'
                   when 'Autob_reg' then '2'
                   when 'Tramvajs' then '3'
                   when 'Trolejbuss' then '4' end) as Transport,
         t1.marsrutas||' *', t2.pavadinimas, idkr as kid, t1.tvarkarastis,
         t3.atributas
    from admin.tTvarkarasciai as t1, admin.tkryptys as t2,
         admin.tTvarkarasciutrukmiuatributai as t3, TAB_TEMPMARSH
    where t1.id = t2.idtvarkarastis and
          t3.idTvarkarastis =  t1.id and t1.id=idmarsh and t2.id=idkr;
```

Izvada piemērs: *tid*= ’546’, *Transport*= ’1’, *marsrutas*= ’22a’, *pavadinimas*= ’Katedrāle – Lidosta “Rīga”’, *kid*= ‘3979’, *tvarkarastis*= ‘5’, *atributas*= ‘6-7’, kas atbilst 22a autobusa maršrutam brīvdienās.

No sākuma kursors atlasa galvenos virzienus, kur A>B. Viens maršruts TAB_MARSHRUTS tabulā būs piemērots abiem virzieniem: gan A>B, gan B>A. Kad visi galvenie maršruti tika atlasīti *getmarsh* procedūra sāk atlasīt papildus maršrutus no TAB_TEMPMARSH tabulas. Lai atdalītu papildus maršrutus no galvenajiem, katram papildus maršruta numuram tika pievienots simbols ‘*’.

Katru ielasītu rindu procedūra saglabā TAB_MARSHRUTS tabula, kā arī katrai izpilda sekojošas darbības: atlasa atbilstoša maršruta punktus, pieturas un laikus. Punktu un pieturu kursors izskatās sekojoši:

```
declare qq_cur dynamic scroll cursor for
  select x, y, pavadinimas, t3.tipas, t3.num, t1.idkryptis,t2.idStotele,
         t2.idAtkarpa
  from admin.tlaikai as t1, admin.tatkarposestoteles as t2
  left outer join admin.tatkarpos as t3 on
    t2.idstotele = t3.id and t3.tipas = 4
  where (t1.idatkarpa = t2.idatkarpa) and ((x is not null) and
    ((t2.nuotolis = 0 or t2.nuotolis is null) and
    (t3.tipas <> 4 or t3.tipas is null) or t3.tipas = 4)) and
    t1.idtvarkarastis = ida and idmasina = minmas and
    t1.idkryptis in(virziens,pretvirziens)
  order by atvyksta asc;
```

Kur *minmas* ir minimālais transportlīdzekļa identifikators no tMasinos tabulas, kas kursē šajā maršrutā; *pretvirziens* – maršruta pretējais virziens, kas atlasās atsevišķā pieprasījumā atkarība no tā vai maršruts ir galvenais vai papildus maršruts. Rezultātā procedūra saņem posma galvenā punkta koordinātes, nosaukumu, skaitļi, kas nosaka punkta tipu (pietura, krustojums vai cits), numuru papildus vajadzībām, punkta virziens (uz kādu galapunktu brauc transportlīdzeklis caur šo punktu), galvena un papildus punktu identifikatorus. tAtkarposeStoteles tabula palīdz atrast visus punktus, bet tAtkarpos tabula atdala pieturas starp tiem. Numurs papildus vajadzībām ir nepieciešams lai atrast pareizas pieturas, tas nozīme, ja piemēram, maršrutam ir pietura Merķeļa iela, tad rezultātā tas paradīsies 4 reizes ar vienādo *num* lauka vērtību, jo reālajā dzīvē ir 4 pieturas ar šo nosaukumu. Visām maršruta pieturam vienā virzienā šim numuram ir jāatšķiras, tātad katrai pieturai procedūra pārbauda vai iepriekšējas pieturas numurs ir vienāds ar tikko nolasīta. Nepieciešamas ir tikai tās, kur šis numurs atšķiras.

Pēc šī kursora datiem tiek aizpildīta TAB_MARSHCOORD tabula. Procedūra ieraksta visus punktus un pieturas, kas neatkārtojas ar iepriekšējo, pareizā secība, kuru kontrolē *place* lauks (pirmajām maršruta punktam tas ir vienāds ar 1, otrajām ar 2 un tā tālāk), pārveido Rīgas Satiksmes x un y koordinātes 1188 koordinātšu sistēmā, kā arī pievieno maršruta identifikatoru katram punktam un pieturai. Punktu nolasīšana un ierakstīšana TAB_MARSHCOORD tabulā notiek kamēr nav izveidots pilns aplis: no pirmā galapunkta līdz šim pašam punktam. Tas nozīme, ka maršruta pirmais un pēdējais punkti būs vienādi.

Trešais kursors ir nepieciešams lai katrai pieturai ierakstīt laiku, bet šajā gadījumā nav tik viennozīmīgi kā maršrutu un punktu gadījumā. Pēc plāna TAB_MARSHTIME tabula satur laiku tikai teksta rindas formātā, lai būtu vieglāk un ātrāk atlasīt tos beigas projektā, bet no datu bāzes viedokļa tas nav racionāli. Pirmkārt lielās grūtības izraisīs laiku maiņa. Ja kāda konkrēta transportlīdzekļa saraksts mainīsies uz vieno un to pašu laika intervālu, tad *getmarsh* procedūru vismaz vienam konkrētam maršrutam būs jāizpilda no jauna. Otrkārt tas neļaus saistīt viena transportlīdzekļa laikus divās dažādas pieturās. Piemēram, uzzināt kad autobuss būs otrā galapunktā, ja no pirmā viņš izbrauca pulksten deviņos. Ņemot vērā šos trūkumus, labāk uztaisīt vēl vienu tabulu, kurā laiki glabāsies time formātā: hh:mm:ss, kur hh nozīme stundas, mm- minūtes un ss- sekundes. Šim mērķim ir uztaisīta TAB_MARSHTIMESTAMPS tabula (sk. Zīmējumu 13.).

Name	Type	Column ID	Allows Nulls
id_marshtimest...	numeric(9,0)	1	No
times	time	2	Yes
marshtime	numeric(9,0)	3	Yes
marshcoord	numeric(9,0)	4	Yes

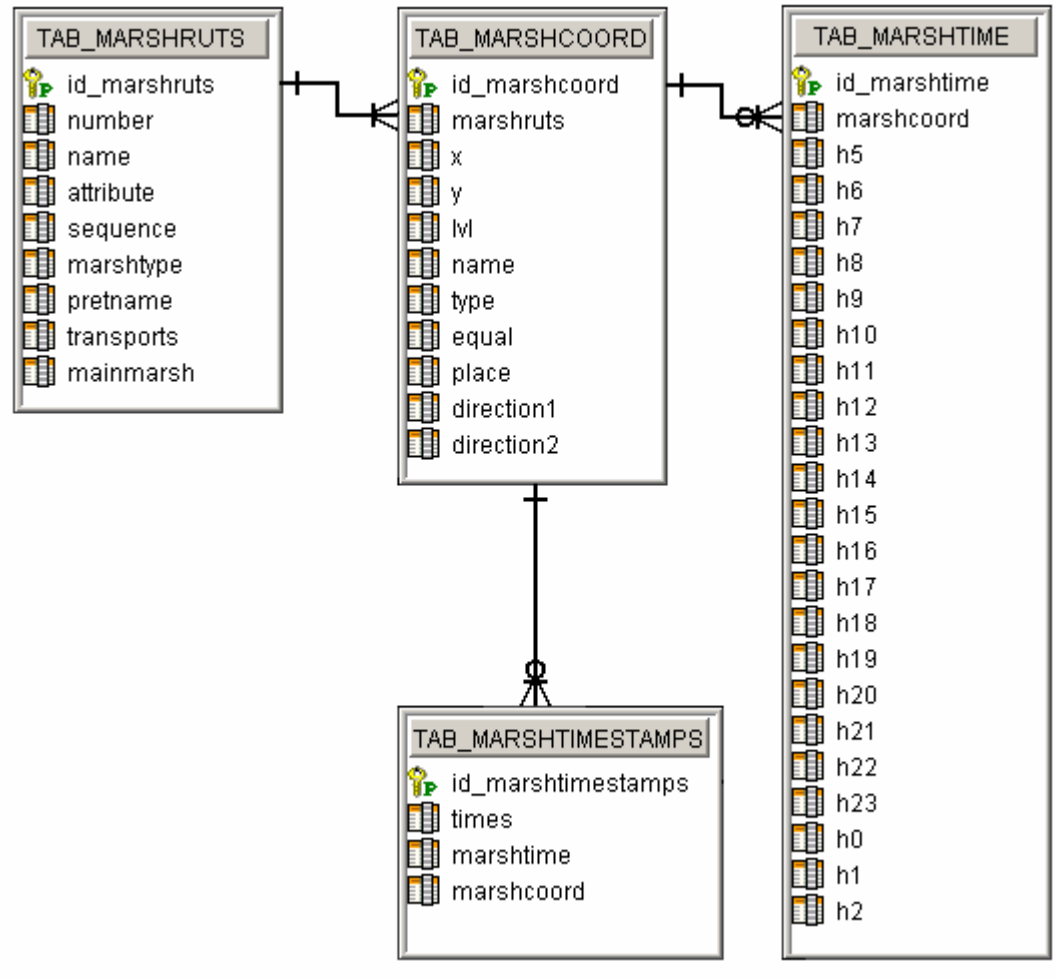
Zīmējums 13. TAB_MARSHTIMESTAMPS

TAB_MARSHTIMESTAMPS satur:

1. *id_marshtimestamps* – tabulas primārā atslēga
2. *times* – laiks formātā hh:mm:ss
3. *marshtime* – TAB_MARSHTIME un TAB_MARSHTIMESTAMPS tabulu saite, identifikators no laiku tabulas
4. *marshcoord* – ārēja atslēga no TAB_MARSHCOORD tabulas, punkta identifikators

Līdz ar jaunās tabulas pievienošanu izmainīsies arī maršrutu tabulu ER modelis.

Papildināta ER modeļa izskatu var apskatīt Zīmējumā 14.



Zīmējums 14. Papildināts ER modelis

Getmarsh procedūras uzdevums ir aizpildīt TAB_MARSHTIMESTAMPS tabulu ar Rīgas Satiksmes tabulu datiem. Tas ir veikts ar sekojoša kursora palīdzību:

```
declare time_cur dynamic scroll cursor for
  select truncnum((t1.atvyksta+t1.Trukme*
   (select ifnull(sum(nuotolis),0,sum(nuotolis))
    from admin.tatkarposestoteles
    where idatkarpa = atkarpa and nuotolis <> 0 and id <=
     (select min(t2.id)
      from admin.tLaikai as t1,admin.tAtkarposeStoteles as t2
      where t1.idtvarkarastis = ida and idstotele = dotid and
            t1.idatkarpa=t2.idatkarpa))/(t3.nuotolis+.000001))/60,0),
    round(mod(t1.atvyksta+t1.Trukme*
     (select ifnull(sum(nuotolis),0,sum(nuotolis))
      from admin.tatkarposestoteles
      where idatkarpa = atkarpa and nuotolis <> 0 and id <=
       (select min(t2.id)
        from admin.tLaikai as t1,admin.tAtkarposeStoteles as t2
        where t1.idtvarkarastis = ida and idstotele = dotid and
              t1.idatkarpa = t2.idatkarpa))/(t3.nuotolis+.000001),60),0)
  from admin.tlaikai as t1,admin.tatkarposestoteles as t2,
       admin.tatkarpos as t3
```

```
where t1.idatkarpa = t2.idatkarpa and t1.idatkarpa = t3.id and
      t2.idstotele = dotid and t1.tipas in(0,8) and t1.idatkarpa = atkarpa and
      idtvarkarastis = ida and t1.idkryptis in(virziens,pretvirziens)
order by atvyksta asc;
```

Kursors atrod ierašanas laikus pieturai *idstotele=dotid*, *idatkarpa=atkarpa*, kur *dotid* un *atkarpa* ir galvenā un papildus punkta identifikatori no iepriekšēja kursora. Kursors izrēķina pareizo pieturas laiku ar formulu: sāklaiks+ilgums*nattālums/attālums, kur sāklaiks ir sākuma laiks, kad transportlīdzeklis nobrauca līdz posma sākumam; ilgums- posma kopējais izbraukšanas ilgums minūtes (tLaikai tabulas *Trukme* lauka vērtība); nattālamus- cik tālu pietura atrodas no posma sākuma, jeb nobrauktais attālums no punkta sākuma; attālums – kopējais posma garums. Kursora rezultāta formāts ir hh:mm, piemēram, 5:30.

Kā bija minēts nodaļā 4.3.2.1. Rīgas Satiksmes datu bāzes apraksts, laiks galapunktiem ir dažāds. Jāzina ne tikai kad transportlīdzeklis nobrauc līdz galapunktam, bet arī kad viņš izbrauc nākamajā braucienā. Katrs galapunkts TAB_MARSHCOORD tabulā ierakstās divas reizes un katrai no tiem jābūt savs saraksts ar laikiem: vienai kad transportlīdzeklis nobrauc līdz pieturai, tātad ierašanas laiki, bet otrai – izbraukšanas laiki. Lai uzzinātu transportlīdzekļa atiešanas laikus no galapunktiem tiek izmantots vēl vien kursors time2_cur, kas ir ļoti līdzīgs aprakstītam time_cur, tomēr tiem ir dažādas atšķirības: visur *Atvyksta* laukā vietā ir *Isvyksta*, tātad kursors atlasa nevis ierašanas, bet izbraukšanas laikus, kā arī sakārtošana notiek pēc *Isvyksta* lauka. Divus kursorus nevar apvienot vienā, kas nolasītu gan *Atvyksta*, gan *Isvyksta* lauku vērtības, jo ir vēl viena nopietna atšķirība. Kad transportlīdzeklis sāk jauno braucienu maršrutā, nevar ierobežot brauciena tipu jeb režīmu tā kā tas bija time_cur: t1.tipas in(0,8). Tā vietā time2_cur ir t1.tipas in (0,2,4,8,9,12,16), tātad jāskatās visi režīmi, izņemot nulles reisus.

Pēc saņemtiem datiem tiek aizpildīta TAB_MARSHTIMESTAMPS tabula.

Ar kursoru palīdzību procedūra daļēji aizpilda TAB_MARSHRUTS, TAB_MARSHCOORD un TAB_MARSHTIMESTAMPS tabulas. Palika aizpildīt šīs tabulas līdz beigām, tas nozīme aizpildīt laukus kuru nebija RS tabulās, kā arī TAB_MARSHTIME tabulu. Šiem mērķiem tika uzrakstītas papildus procedūras, kuru izsaukšana notiek *getmarsh* procedūras beigās.

```
call internet.getcoordlevels(marshid)
...
call internet.getvienpiet();
call internet.gettimes();
call internet.addmarshsequence();
call internet.addmarshtypes();
call internet.getdirections();
```

Kur internet ir procedūru izveidotāja lietotāja vārds. *Getmarsh* procedūras tekstu var apskatīt pielikuma 8.punktā – GETMARSH procedure. Tālāk ir dots šo papildus procedūru apraksts.

Getcoordlevels procedūra izsaucās katram maršrutam atsevišķi ciklā. Šī procedūra saņem vienu parametru- maršruta identifikatoru. *Getcoordlevels* mērķis ir TAB_MARSHCOORD tabulas *lvl* lauka aizpildīšana, kas praktiski nozīmē katram maršruta punktam noradīt kartes līmeni, no kura tas sāk rādīties interaktīva kartē. Jo kartes līmenis ir precīzāks, jo vairāk punktu jārāda, lai maršruta līnija būtu vieglāk uztverama. Procedūra skaita attālumus no punkta līdz nākamiem punktiem un līdz ar to ieraksta atbilstošas vērtības tabulā. Vislielākā nultajā mērogā attālumam jābūt lielākam par 37500 pikseļiem, jeb 15000 metriem, pirmajā – lielākam par 7500 pikseļiem, jeb 3000 metriem, otrajā – lielākam par 1500 pikseļiem, jeb 600 metriem, trešajā - atbilstoši 1500 pikseļiem, jeb 120 metriem, ceturtajā – 300 pikseļiem, jeb 24 metriem, piektajā attālumiem nav nozīmes. Procedūra

izmanto kursoru, kas atlasa visus maršruta punktus, tā identifikatoru un koordinātes. Lai būtu skaidrāk aplūkosim 40 autobusu Ziepniekkalns – Jugla 3 (*id_marshruts*=56). Pēc procedūras izpildīšanas šim maršrutam situācija ar līmeņiem ir sekojoša:

5 līmenis – 296 punktiem;

4 līmenis – 442 punktiem;

3 līmenis – 254 punktiem;

2 līmenis – 46 punktiem;

1 līmenis – 8 punktiem;

Lauku ar *lvl*=0 nav.

Kartes konkrētajā līmenī būs parādīti visi punkti kur *lvl* ir mazāks vai vienāds ar kartes līmeni. Tātad piektajā līmenī 40 autobusam būs 1046 punkti (296+442+254+46+8= 1046) un maršruta līnija būs ļoti precīza, bet visgalvenākā līmenī nebūs neviena punkta.

Getcoordlevels procedūras tekstu var apskatīt pielikuma 9.punktā – GETCOORDLEVELS procedure.

Pēc *getmarsh* procedūras izpildes *getcoordlevels* procedūra būs izpildīta visiem maršrutiem, tātad līmenis būs noradīts visiem punktiem. Paliek tikai neskaidrības ar pieturām. Nav labi, ja viena pietura radīsies visos līmeņos, bet cita tikai visprecīzākā līmeni, tāpēc tika nolemts visām pieturām piešķirt vienādu līmeni. Līdz ar to pēc cikla *getmarsh* procedūrā ir pievienota rinda:

```
update admin.tab_marshcoord set lvl = 3 where  type = 4;
```

Visām pieturām TAB_MARSHCOORD tabulas *type* lauks ir vienāds ar 4. Tas saglabājas no Rīgas Satiksmes datu bāzes (tAtkarpos tabulas *Tipas* lauks). Pēc vaicājuma izpildes visām pieturām līmenis būs vienāds ar 3. Tas nozīmē, ka visas maršrutu pieturas kartē radīsies sākot no 3 līmeni, tātad: trešajā, ceturtajā un piektajā līmeņos.

Nākamā procedūra ir *getvienpiet*. Šī procedūra ir nepieciešama lai atrast vienādas pieturas un ierakstīt tām vienādu TAB_MARSHCOORD tabulas *equal* lauku. Šis lauks ir nepieciešams dažu uzdevumu risināšanai, ka visgalvenāko var uzskatīt viena maršruta pieturu saite dažādās dienās. Tā, piemēram, ja maršruts kursē darba dienās un brīvdienās, zinot maršruta numura, transporta veida un *equal* lauka vērtības var uzzināt kāda brīvdienas maršruta pietura atbilst darba dienas maršruta pieturai. Tas ir nepieciešams lai gatavā projektā maršrutu tabulā būtu iespējams radīt tikai vienu maršrutu, kuram varētu pārslēgt kursēšanas dienas lai redzētu laika sarakstus. Ir vēl viena iespēja, kuru var realizēt zinot kuras pieturas ir vienādas. Katrai pieturai var uzzināt visus maršrutus, kuri pietur šajā punktā, vienkārši atlasot visas TAB_MARSHCOORD rindas, kur *equal* ir vienāds ar kādu konkrētu skaitļi.

Pār vienādām pieturām ir nolemts uzskatīt pieturas, kurām sakrīt nosaukumi, bet koordinātes atšķiras ne vairāk par 100 pikseļiem – 40 metriem, tas attiecās gan uz x, gan uz y koordināti. Procedūra ar kursora palīdzību nolasa visu pieturu sarakstu (TAB_MARSHCOORD.type = 4). Tas ir sakārtots pēc pieturu nosaukumiem un koordinātēm (TAB_MARSHCOORD.name, TAB_MARSHCOORD.x, TAB_MARSHCOORD.y). Ciklā notiek pārbaude:

```
if prevpietname = pietname and abs(prevpietx-pietx) < 100 and
   abs(prevpiety-piety) < 100 then
  update admin.tab_marshcoord set equal = i where id_marshcoord = ida
else
```

```
  set i=i+1;
  update admin.tab_marshcoord set equal = i where id_marshcoord = ida;
  set prevpietname=pietname;
  set prevpietx=pietx;
  set prevpiety=piety
end if
```

Kur *prevpietname*, *prevpietx*, *prevpiety* ir iepriekšējas pieturas dati; *ida*, *pietname*, *pietx*, *piety* – tikko nolasītas pieturas identifikators (*id_marshcoord*), nosaukums un koordinātes; i – skaitītais, kas kontrolē kādu vērtību jāraksta *equal* laukā pašreizējā momentā. Var redzēt ka *getvienpiet* procedūra pārbauda vai pieturas nosaukums ir vienāds ar iepriekšējo un x un y koordinātes atšķiras ne vairāk par 100 pikseļiem un līdz ar salīdzināšanas datiem vai ieraksta šai pieturai iepriekšējas pieturas *equal* lauka vērtību, vai palielina skaitītāju un ieraksta *prevpietname*, *prevpietx* un *prevpiety* nolasīta punkta datus. Pēc *getvienpiet* izpildes lauks equal būs aizpildīts visām pieturām, neskatoties vai tas ir galvenais, vai papildus maršruts. *Getviepiet* procedūras tekstu var apskatīt pielikuma 10.punktā – GETVIENPIET procedure.

Gettimes ir procedūra kas ir nepieciešama tieši TAB_MARSHTIME tabulas aizpildīšanai, kas notiek saskaņā ar TAB_MARSHTIMESTAMPS tabulas datiem. Procedūra satur divus kursorus, pirmais atlasa visas pieturas, otrais- atbilstošas pieturas laikus noteiktajā stundas intervālā, piemēram pulksten piecos no rīta.

Nolasot pieturas identifikatoru (*id_marshcoord*), procedūra atlasa visus pieturas laikus pēc kārtas, kur *hour(times) = hhour*, kur *times* ir laiks no TAB_MARSHTIMESTAMPS tabulas, *hhour* ir skaitītais kas automātiski palielinās ciklā, *hour* – Sybase iebūvēta procedūra kas no laika formātā hh:mm:ss atdala hh. Pie tam kursors atlasa tikai laika minūtes. Atrasta laika minūtes procedūra pievieno teksta rindai, kas anulējas kad palielinās *hhour* mainīga vērtība. Kad visi laiki šajā intervālā ir pievienoti teksta rindai, procedūra ieraksta iegūto rindu TAB_MARSHTIME atbilstošajā laukā *h**n***, kur ***n***=*hhour* un *marshcoord* ir vienāds ar nolasītas pieturas identifikatoru, piemēram:

```
if hhour = 6 then
  update admin.tab_marshtime set h6 = minutest where id_marshtime = i;
  set hhour=hhour+1
else
```

Kur *minutest* – teksta rinda, kur glabājas minūtes formāta ‘mm1 mm2 ...’, *i* – pieturas identifikators (*id_marshcoord*). Procedūra pati izveido nepieciešamas rindas TAB_MARSHTIME tabulā cikla sākumā. Kad visi lauki pieturai ir noteikti, procedūra nolasa nākamo pieturu. *Gettimes* procedūras tekstu var apskatīt pielikuma 11.punktā – GETTIMES procedure.

Pēc *gettimes* procedūras izpildes visām pieturām eksistē atbilstoši lauki TAB_MARSHTIME tabulā un visi lauki (h5, h6, h7, h8, h9, h10, h11, h12, h13, h14, h15, h16, h17, h18, h19, h20, h21, h22, h23, h0, h1) ir aizpildīti ar laikiem saskaņā ar TAB_MARSHTIMESTAMPS tabulas datiem. Dažos gadījumos tie ir tukši, ja šajā laikā maršrutam nav braucienu.

Addmarshsequence aizpilda TAB_MARSHRUTS tabulas *sequence* lauku, kas praktiski nozīmē pārveido maršruta numuru skaitliskā formā. Šis lauks ir nepieciešams lai varētu pareizi sakārtot maršrutu sarakstu gatavā projektā, jo teksta rindas sakārtošana notiek pa atsevišķiem simboliem, tas nozīmē, ka no sākuma būs visi maršruti, kur pirmais cipars ir 1 (1, 11, 12, 13, ..), pēc tam tie, kuru otrais simbols ir 2: 2, 21, 22, .. un tā tālāk. *Sequence* lauks atļaus sakārtot pēc numuru skaitliskām vērtībām: 1, 2, 3, .. .

Procedūra izmanto kursoru, kas nolasa visu maršrutu identifikatorus un numurus. Lai

pārveidot numuru *addmarshsequence* izmanto *normalizetxt* funkciju (sk. nodaļā 4.2.1.1. Teksta pārveidošanas funkcija) un nomaina numura visus burtus un pieturzīmes uz tukšumiem. Pēc pārveidošanas šo teksta rindas mainīga vērtību pārtaisa uz skaitli (integer) ar iebūvētas *convert* funkcijas palīdzību. Iegūta numura skaitliskā formā vērtību procedūra ieraksta TAB_MARSHRUTS tabulas atbilstošajā laukā. *Addmarshsequence* procedūra apstrādā visus maršrutus pēc kārtas, tāpēc pietiek izsaukt to tikai vienu reizi *getmarsh* procedūrā. *Addmarshsequence* procedūras tekstu var apskatīt pielikuma 12.punktā – ADDMARSHSEQUENCE procedure.

Nākama procedūra ir *addmarshtypes*. Šī procedūra ieraksta *marhstype* laukus TAB_MARSHRUTS tabulā. Tas nozīmē: visiem maršrutiem ar vienādo numuru, bet dažādām kursēšanas dienām atrod tipu, kas nosaka, kad vispār šis maršruts ir pieejams un kurās dienās maršrutam sakrīt laika saraksts Šis tips būs nepieciešams laiku tabulas veidošanai gatavā projektā lai pārslēgtu laika režīmus.

Maršrutu atlase notiek ar kursora palīdzību, kas atrod visu maršrutu identifikatorus (TAB_MARSHRUTS.id_marshruts), numurus (TAB_MARSHRUTS.number) un transportlīdzekļa veidus (TAB_MARSHRUTS.transports). Lai atrastu maršruta tipu procedūra saskaita cik reizes TAB_MARSHRUTS tabulā atkārtojas maršruts ar šo pašu numuru un *transports* lauka vērtību un noskaidro kādas tiem ir *attribute* lauka vērtības. Iegūto skaitļi procedūra ieraksta tabulā:

```
update admin.tab_marshruts set marshtype = tips where id_marshruts = ida;
```

Kur *tips* ir iegūtais skaitlis, kas nosaka transportlīdzekļa laika tipu; *ida*- pēdēja nolasīta maršruta identifikators. Visas darbības procedūra atkārto katram maršrutam ciklā, kamēr visi maršruti netiek apstrādāti. *Addmarshtypes* procedūras tekstu var apskatīt pielikuma 13.punktā – ADDMARSHTYPES procedure.

Getdirections ir pēdēja papildus procedūra, kas palīdz aizpildīt *direction1* un *direction2* laukus TAB_MARSHCOORD tabulā. *Direction1* lauks ir nepieciešams lai izveidot divas dažādas pieturu sarakstus maršrutam dažādos virzienos, kā arī palīdz atrast transportlīdzekļa atiešanas laikus. *Direction2* ļauj atlasīt tikai maršruta galapunktus. Viens no šī lauka uzdevumiem ir iespējamība radīt maršrutu galapunktus visos karšu līmeņos.

Procedūra nolasa maršruta identifikatorus (TAB_MARSHRUTS.id_marshruts) ar kursoru palīdzību. Maršrutu nolasīšana un apstrāde notiek ciklā. Katra maršruta vidējai pieturai- otrajam galapunktam- *direction2* ir vienāds ar 1, kas tiek piešķirts galvenajā *getmarsh* procedūrā. Šo īpašību izmanto *getdirections* procedūra, kas atrod šo vidējo punktu un visiem maršruta punktiem un pieturām, kur identifikators ir mazāks par atrasto *direction1* nomaina uz vieninieku, bet pretēja gadījumā uz divnieku. Tas notiek sekojoši:

```
select min(id_marshcoord) into minid
  from admin.tab_marshcoord
  where marshruts = marsh and direction2 = 1;
update admin.tab_marshcoord set direction1 = 1
  where id_marshcoord < minid and marshruts = marsh and direction1 is null;
update admin.tab_marshcoord set direction1 = 2
  where id_marshcoord >= minid and marshruts = marsh and direction1 is null;
```

Kur *minid* ir mainīgais, kur glabāsies maršruta otrā galapunkta identifikators, *marsh*- nolasīta maršruta identifikators. Tādejādi tiek aizpildīts *direction1* lauks. Lai aizpildītu *direction2* lauku līdz galam *getdirections* procedūra ieraksta maršruta pirmajai pieturai (galapunktam) *direction2*=1, bet visiem pārējiem *direction2*=0:

```
update admin.tab_marshcoord set direction2=1
  where id_marshcoord=
        (select min(id_marshcoord) where type=4 and marshruts=marsh);
update admin.tab_marshcoord set direction2=0
  where marshruts=marsh and direction2 is null;
```

Pēc procedūras izpildes *direction1* un *direction2* lauki ir aizpildīti visiem punktiem. Visas nepieciešamas *getdirections* procedūras darbības varētu pievienot *getmarsh* procedūrai un iztikt bez šīs papildus procedūras, bet tad būtu grūtāk ieviest kaut kādas izmaiņas, kas ir saistītas ar virzieniem. *Getdirections* procedūras tekstu var apskatīt pielikuma 14.punktā – GETDIRECTIONS procedure.

Palika jautājums par papildus un galveno maršrutu saiti, tas nozīmē katram papildus maršrutam atrast galveno maršrutu, lai gatavā projektā būtu iespējams izvēlot vienu no galvenajiem maršrutiem redzēt kādi tam ir pieejami papildus maršruti. Šim mērķim *getmarsh* procedūrās beigās ir uzrakstītas sekojošas rindas:

```
update admin.tab_marshruts pmarsh set pmarsh.mainmarsh=
  (select min gmarsh.id_marshruts
     from admin.tab_marshruts gmarsh
     where gmarsh.number=pmarsh.number||' *' and
           gmarsh.transports=pmarsh.transports)
where pmarsh.number like '%*%' and mainmarsh is null
```

Tas ļauj visiem papildus maršrutiem uzreiz aizpildīt TAB_MARSHRUTS tabulas *mainmarsh* lauku. Galvenajiem maršrutiem šis lauks paliek tukšs, tas nozīmē, ka *mainmarsh*=null.

Līdz ar to *getmarsh* procedūra, kas atbild par datu kopēšanu no Rīgas Satiksmes tabulām un to pārveidošanu nepieciešamā formātā, ir pabeigta un mērķis ir sasniegts- visas maršrutu tabulas tika aizpildītas.

4.3.3. Papildus iespēju aplūkošana un to daļēja realizācija

Izveidotā datu struktūra ļauj apskatīt konkrētus maršrutus kartē. Lietotājs var izvēlēties maršrutu no dotā saraksta un apskatīt to attēlojumu interaktīvajā kartē. Tomēr būtu ērtāk, ja cilvēks varētu pēc diviem punktiem uzzināt visus iespējamos maršrutus, kā viņš var nokļūt no pirmā punkta līdz otrajam, ieskaitot gadījumus, kad nav iespējams nobraukt šo nogriezni ar vienu transportlīdzekli. Pie tam 'punkts' šajā kontekstā nozīmē ne tikai pieturu, bet jebkuru ģeogrāfisko objektu.

Tātad galvenā no papildus iespējām ir optimālā maršruta meklēšana pēc diviem punktiem. Vissvarīgākais jautājums ir: ko uzskatīt par optimālo maršrutu. Pirmais, kas jāņem vērā, ir pārsēšanos skaits. Vislabāk būtu, ja no viena punkta līdz otram varētu nokļūt ar vienu transportlīdzekli. Otrais- ceļam patērētais laiks.

Šajā nodaļā apskatīsim maršruta meklēšanas piemēru starp pieturām ar vismazāko pārsēšanos skaitu. Šim mērķim ir izveidota jaunā *findroute* funkcija, kas saņem 3 parametrus: *int1* numeric(9), *int2* numeric(9), *tips* integer, kur *int1* un *int2* ir sākuma (tālāk tekstā apzīmēts ar A) un beigu (tālāk tekstā apzīmēts ar B) pieturu identifikatori (TAB_MARSHCOORD.id_marshcoord) un *tips* - skaitlis, kas apzīmē nedēļas dienu: 1- jebkura darba diena, 6- sestdiena, 7- svētdiena. Šis skaitlis ir nepieciešams lai atrastu pareizus maršrutu identifikatorus- viena maršruta numura identifikatori dažādās kursēšanas dienās atšķiras. Funkcijas uzdevums atgriezt maršruta identifikatoru, ar kuras palīdzību var nokļūt no punkta A līdz punktam B, bet, ja tāds neeksistē,

atrast divus maršrutus, kas ļauj nokļūt līdz nepieciešamajam punktam, un pārsēšanas pieturas identifikatoru.

Jāņem vērā, ka vienas pieturas identifikatori dažādiem maršrutiem atšķiras, tāpēc pareizāk skatīties nevis *id_marshcoord* lauku, bet *equal* lauku, kas savieno vienādas pieturas. Tādēļ pirmā funkcijas darbība ir atbilstošo punktu *equal* lauku atrašana. Tas notiek atsevišķos pieprasījumos. Rezultāti tiek ierakstīti uz jaunajiem mainīgiem.

Tālāk notiek maršruta meklēšana. Atbilstoši trešā parametra vērtībai *findroute* funkcija izpilda vienu no pieprasījumiem. Apskatīsim pieprasījumu, kad *tips*=6:

```
select top 1 m1.marshruts into marshr
  from admin.tab_marshcoord as m1,admin.tab_marshcoord as m2,
       admin.tab_marshruts as s1
  where m1.equal = point1 and m2.equal = point2 and
        m1.marshruts = m2.marshruts and m1.marshruts = s1.id_marshruts and
        s1.attribute in('6','6-7','1-7')
```

Kur *marsh* – mainīgais, uz kuru tiek ierakstīts atrasta maršruta identifikators (ja tāds eksistē); *point1* – pieturas A *equal* lauks, *point2* – pieturas B *equal* lauks. Pieprasījums atgriež pirmā maršruta identifikatoru, kas savieno punktus A un B un kursē nepieciešamajā dienā.

Ja maršruts netiek atrasts un *marshr* mainīgā vērtība ir vienāda ar ‘’, tad jāmeklē divi maršruti un to kopīgo punktu jeb pārsēšanas pieturu.

Lai maršrutu meklēšana būtu vieglāka un funkcijas izpildīšanas laiks būtu pēc iespējas mazāks, vajag vienkāršot kopīgo pieturu meklēšanu, ja ir zināmi divu maršrutu identifikatori. Šī uzdevuma risināšanai tika izveidota papildus tabula- TAB_MARSHRUTS_SEQUENCE (sk. Zīmējumu 15.).

Name	Type	Column ID	Allows Nulls
id_marshruts_sequence	integer	1	No
marshruts1	integer	2	No
marshruts2	integer	3	No
sequence	integer	4	Yes

Zīmējums 15. TAB_MARSHRUTS_SEQUENCE

TAB_MARSHRUTS_SEQUENCE kolonas:

5. *id_marshruts_sequence* – tabulas primārā atslēga
6. *marshruts1* – ārējā atslēga no TAB_MARSHRUTS tabulas, pirmā maršruta identifikators
7. *marshruts2* – ārējā atslēga no TAB_MARSHRUTS tabulas, otrā maršruta identifikators
8. *sequence* – maršrutu kopējā punkta *equal* lauks

TAB_MARSHRUTS_SEQUENCE tabula saturēs visus maršrutus. Tātad, lai uzzinātu, vai maršrutiem ir kopīga pietura, pietiks atrast atbilstošas rindas *sequence* lauku.

Tabulas *marshruts1* un *marshruts2* lauku aizpildīšana notiek sekojoši:

```
insert into tab_marshruts_sequence (marshruts1,marshruts2)
```

```
select m1.id_marshruts, m2.id_marshruts
  from tab_marshruts m1, tab_marshruts m2
  where m1.id_marshruts < m2.id_marshruts
  order by m1.id_marshruts, m2.id_marshruts;
commit;
```

Identifikatori (*id_marshruts_sequence*) tiek ierakstīti automātiski. Lai aizpildītu *sequence* lauku, ir uzrakstīts vēl viens pieprasījums:

```
update tab_marshruts_sequence set sequence=
  (select top 1 equal
   from tab_marshcoord mm1
   where marshruts=marshruts1 and type is not null and  equal in
    (select distinct equal
     from tab_marshcoord mm2
     where marshruts=marshruts2 and type is not null));
commit;
```

kas atrod pirmo kopējo pieturu diviem maršrutiem un ieraksta to TAB_MARSHRUTS_SEQUENCE tabulas atbilstošajā rindā. Ja kopīgu punktu nav, tad *sequence* lauks maršrutiem paliek tukšs.

Pēc insert un update izpildes tabula ir aizpildīta visiem maršrutiem. Ar tās palīdzību var viegli uzzināt, kā saistās maršruti un kā no viena maršruta var pārsēsties uz kādu citu.

Findroute funkcija atrod visus maršrutus, kas pietur punktā A, un visus maršrutus, kas pietur punktā B. Tā kā kopīga maršruta starp sarakstiem nav (ja būtu, funkcija neizpildītu šīs darbības vispār), funkcija atrod pirmo kopīgo punktu diviem maršrutiem, kur maršruti ir patvaļīgi maršruti no diviem dažādiem sarakstiem. Pieturas meklēšana notiek ar jaunās TAB_MARSHRUTS_SEQUENCE tabulas palīdzību.

Atbilstoši *tips* mainīgā vērtībai tālākās darbības mazliet atšķiras. Apskatīsim gadījumu, kad *tips*=6:

```
select distinct marshruts into #pointmarsh1
  from admin.tab_marshcoord,admin.tab_marshruts
  where equal = point1 and marshruts = id_marshruts and
        attribute in('6','6-7', '1-7');
select distinct marshruts into #pointmarsh2
  from admin.tab_marshcoord,admin.tab_marshruts
  where  equal = point2 and marshruts = id_marshruts and attribute in('6','6-7',
'1-7')
...
select top 1 p1.marshruts || ' ' || p2.marshruts || ' pietura ' || sequence
  into retstring
  from admin.tab_marshruts_sequence,#pointmarsh1 as p1,#pointmarsh2 as p2
  where p1.marshruts = marshruts1 and p2.marshruts = marshruts2 and
        sequence is not null
```

Kur *#pointmarsh1* un *#pointmarsh2* ir papildus tabulas, kuras eksistē tikai procedūras funkcijas izpildes laikā; *retstring* – rezultāts, formātā: id1 id2 pietura seq1. id1 un id2 ir maršrutu identifikatori, seq1 – kopīgais maršrutu punkts, pārsēšanās pieturas *equal* lauks. *Findroute* funkcijas tekstu var apskatīt pielikuma 15.punktā – FINDROUTE function.

Funkcija *findroute* atgriež vai nu *marsh* mainīgā vērtību, ja tā nav tukša, vai *retsring*

vērtību pretējā gadījumā. Funkcijas izsaukšanas piemēri:

```
select internet.findroute(8588,9009,1);
select internet.findroute(8588,9009,6);
select internet.findroute(50022,255271,1);
```

Kur: 8588 – pietura Sarkandaugava, 9009 – pietura Veikals "Pļavnieki", 50022 – pietura Rudens iela, 255271 – pietura Graudu iela.

Rezultāti ir sekojoši:

'10' – nozīmē, ka ar maršrutu ar identifikatoru 10: 48 autobuss Ulbrokas kapi – Sarkandaugava var nokļūt no Sarkandaugavas pieturas līdz Veikala "Pļavnieki" pieturai jebkurā darba dienā;

'11' – nozīmē, ka ar maršrutu ar identifikatoru 11: 48 autobuss Ulbrokas kapi – Sarkandaugava var nokļūt no Sarkandaugavas pieturas līdz Veikala "Pļavnieki" pieturai sestdienā;

'59 308 pietura 2994' nozīmē, ka no Rudens ielas līdz Graudu ielai nevar nokļūt ar vienu transportlīdzekli, bet var no sākuma braukt ar 3 autobusu Pļavnieki – Birzes iela līdz pieturai ar *equal* lauku vienādu ar 2294 – Centrālā stacija, bet pēc tam pārsēsties uz 24 trolejbusu (*id_marshruts*=308) Pētersalas iela – a/s "Dzintars".

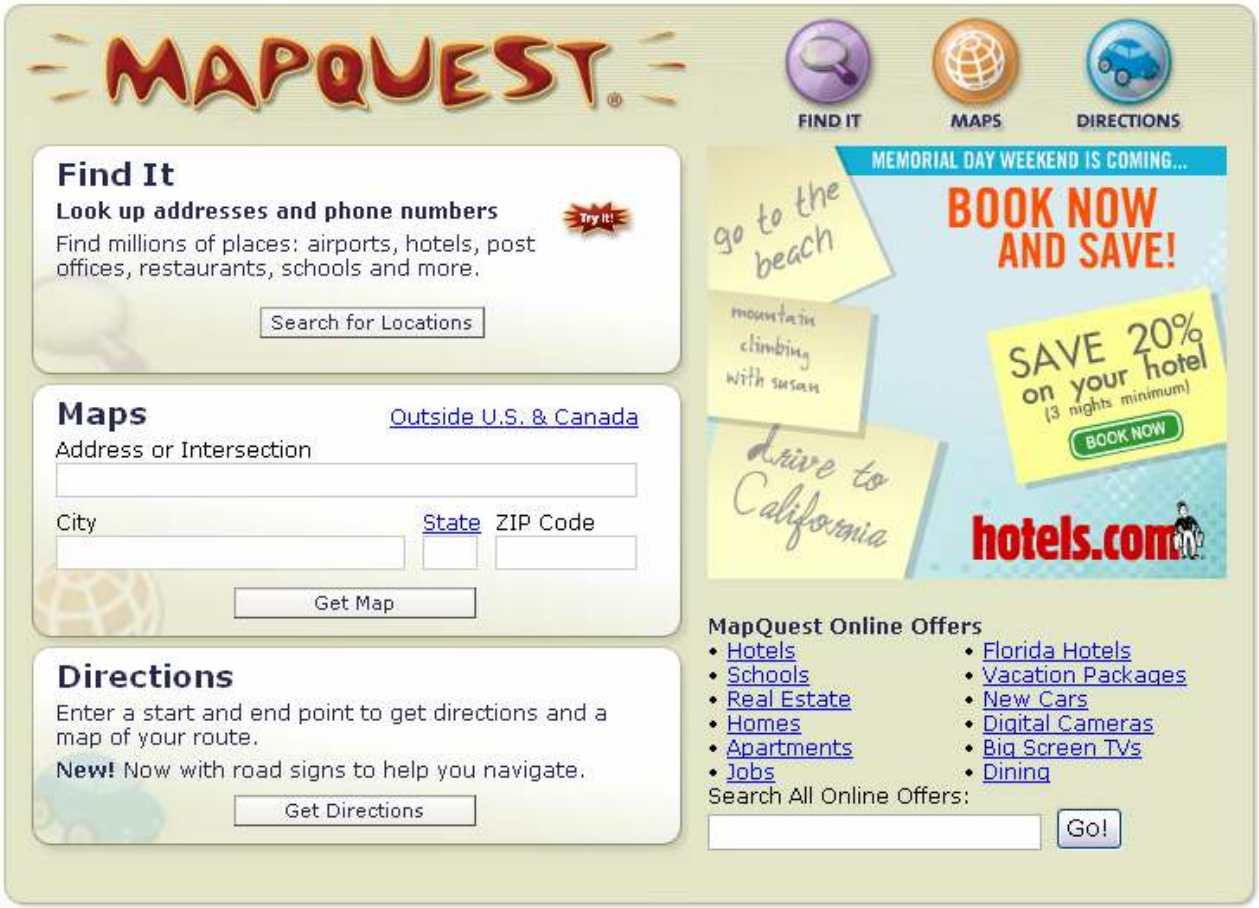
Ar to primārā maršruta meklēšana ir pabeigta. *Findroute* funkcija neņem vērā braukšanas ilgumu, tāpēc atrastais ceļš var nebūt optimāls no laika viedokļa.

5. 1188 kataloga salīdzinājums ar citiem līdzīgiem Interneta katalogiem

Datu bāze tika pārveidota un jaunās iespējas tika realizētas līdz galam. Pastāv iespēja meklēt kartes fragmentu pēc uzņēmuma nosaukuma, kā arī apskatīt uzņēmumu atrašanas vietas un pilsētas un piepilsētas maršrutu līnijas Latvijas interaktīvajā kartē. Šīs funkcijas ir pieejamas dažādos portālos: iespējas, kas saistās ar uzņēmumiem, visur, kur ir pieejams 1188 uzņēmumu katalogs Internetā un pēdējā - Rīgas Satiksmes mājas lapā. Līdz ar to salīdzināšana ar līdzīgiem katalogiem dalās divās daļās: 1188 uzņēmumu kataloga Interneta salīdzināšana ar analogiem un Rīgas Satiksmes mājas lapas Internetā kartes nodaļas apskatīšana un salīdzināšana.

Runājot par 1188 uzņēmumu katalogu, var apgalvot, ka Internetā var atrast ļoti daudz gan Latvijas, gan ārvalstu resursu ar līdzīgu tematiku un meklēšanas iespējām. Eksistē lappuses, kur ir pieejams tikai uzņēmumu katalogs vienā noteiktā teritorijas apgabalā (piemēram, valsts, pilsēta), kā arī ir tādi, kur var apskatīties kartes, vai arī meklēt kaut kādus objektus šinīs kartēs. Lielāko interesi izraisa tie, kuri piedāvā visas iepriekšminētās iespējas.

Sākumā apskatīsim vienu no tiem: http://www.mapquest.com. Mapquest ir Amerikas Savienoto Valstu uzņēmumu katalogs, kura galvenā funkcija sakrīt ar 1188 uzņēmumu kataloga funkciju: glabāt un sniegt apmeklētājiem informāciju par uzņēmumiem. Ērtībai katalogam ir pieejama arī interaktīvā karte.



Zīmējums 16. ASV uzņēmumu katalogs Internetā

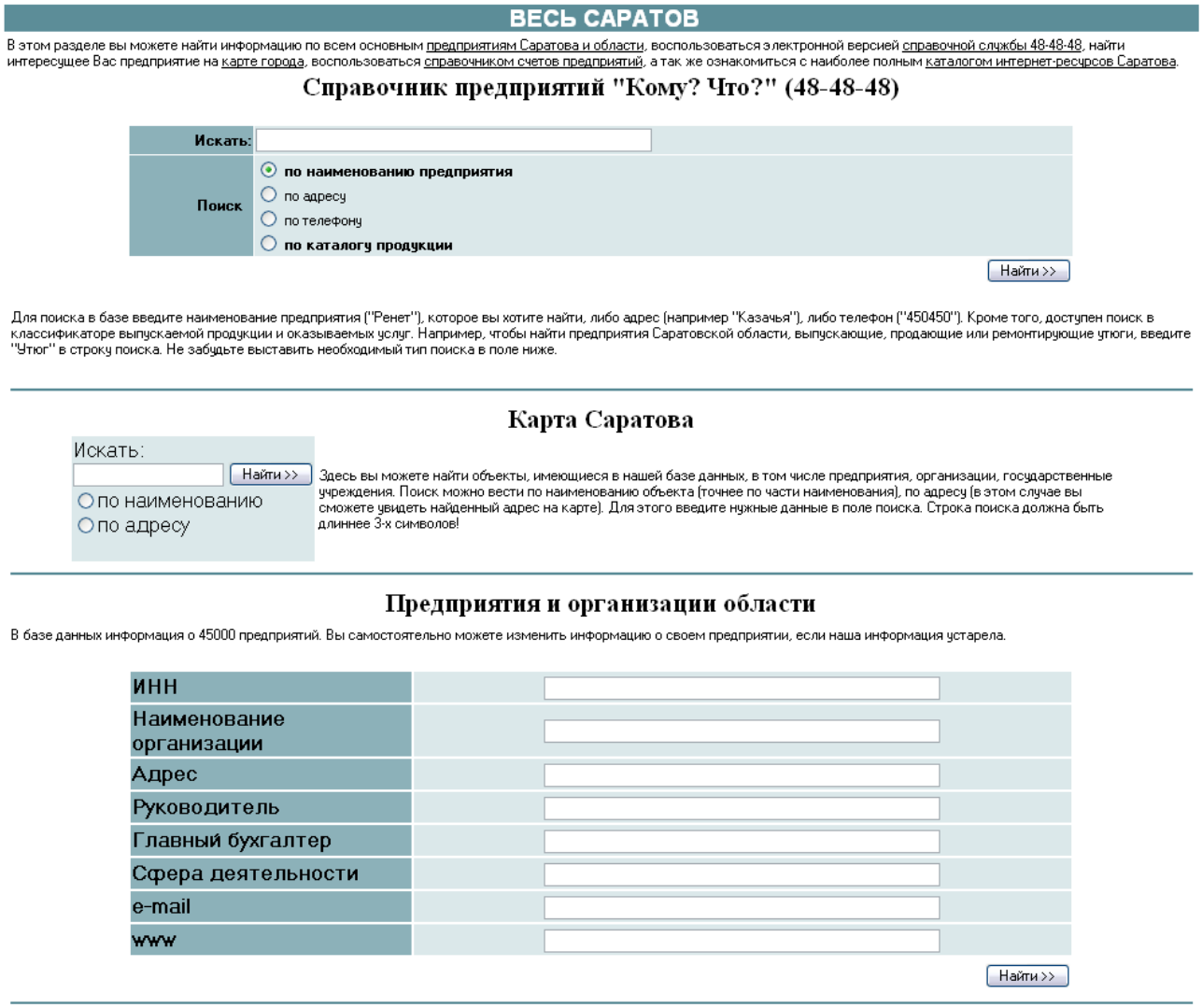
Zīmējumā 16. ir parādīts Mapquest meklētāja izskats.

Pirmā šī kataloga iespēja ir: meklēt lidostu, viesnīcu, pasta nodaļu, restorānu, vidusskolu un citu uzņēmumu adreses un telefona numurus. Meklēt var tikai tos objektus, kas atrodas Amerikas Savienotajās Valstīs. Informāciju var meklēt, ievadot uzņēmuma nosaukumu, adresi, pilsētas nosaukumu, štata nosaukumu vai pasta indeksu. Pie tam pastāv iespēja saglabāt kādu adresi (piemēram, savu mājas vai darba adresi) un meklēt šai adresei tuvākos uzņēmumus.

Objektus kartēs var meklēt pēc dažādiem kritērijiem: adrese, krustojums, pilsēta, štats vai pasta indekss. Svarīga šīs lappuses īpatnība ir ļoti liels karšu krājums, līdz ar to objektus var meklēt ne tikai ASV un Kanādā, bet arī ārpus šo valstu robežām. Pastāv iespēja apskatīt vairāk kā 200 valstu kartes. Valstu sarakstā ir gan lielas valstis, gan ļoti mazas. Diemžēl vairākās valstīs var atrast tikai lielu pilsētu atrašanas vietas [15].

Salīdzinot aprakstīto sistēmu ar 1188 uzņēmumu katalogu, var teikt, ka Mapquest ir papildus iespējas, kā, piemēram, objektu meklēšana vairākās valstīs un adrešu saglabāšana, kas ļauj meklēt tuvākos objektus. Tomēr sistēmai ir arī savi mīnusi. Piemēram, katalogā nevar meklēt informāciju par cita tipa uzņēmumiem: veikaliem, degvielas uzpildes stacijām un citiem. Otrā trūkstošā iespēja ir tā, ka nevar uzreiz meklēt uzņēmumu kartē (kartē var meklēt tikai pēc adreses). Kā arī nevar redzēt atrasto uzņēmumu atrašanās vietu uzreiz kartē.

Vēl viens ārvalstu uzņēmumu katalogs Internetā ir pieejams lappusē: http://www.sarbc.ru/saratov (kataloga izskatu sk. Zīmējumā 17.)



Zīmējums 17. Saratovas uzņēmumu katalogs Internetā

Šis Interneta resurss arī ir ļoti līdzīgs 1188 uzņēmumu katalogam. Šī lappuse ir 48-48-48 uzziņu dienesta elektroniskā versija. Uzņēmumu katalogā var atrast informāciju par Saratovas un tās reģiona uzņēmumiem, kā arī atrast uzņēmuma atrašanas vietu kartē un iepazīties ar Saratovas Interneta resursu pilno katalogu.

'Kam? Kas'(48-48-48) uzņēmumu rokasgrāmatā var meklēt uzņēmumus pēc dažādiem kritērijiem. Lai meklētu bāzē, var ievadīt vajadzīgā uzņēmuma nosaukumu, adresi vai telefona numuru. Kā arī ir pieejama meklēšana pēc izlaistās produkcijas un piedāvāto pakalpojumu klasifikatora. Pie tam, kā jau bija teikts agrāk, meklēšana notiek tikai noteiktā teritorijas apgabalā: Saratovā un tās apkārtnē.

Otrā uzņēmumu kataloga meklēšanas iespēja ir 'Saratovas karte'. Šeit var atrast datu bāzē esošos objektus, tai skaitā uzņēmumus, organizācijas un valsts iestādes. Kā meklēšanas kritērijs var kalpot nosaukums (precīzāk – nosaukuma daļa), adrese (šajā gadījumā kartē tiek parādīta atrastā adrese). Meklēšanas rindas minimālais garums ir 3 simboli. Diemžēl karte ir ļoti neprecīza.

Pēdējā iespēja ir 'Pagasta uzņēmumi un organizācijas'. Uzņēmuma datu bāze glabājas informācija par vairāk nekā 45000 uzņēmumiem. Apmeklētājs pats var atjaunot informāciju par savu uzņēmumu, ja kataloga informācija ir novecojusi. Šinī nodaļā apmeklētājs var meklēt, ievadot sekojošu informāciju par uzņēmumu: uzņēmuma nodokļu maksātāja identifikācijas numuru,

uzņēmuma nosaukumu, adresi, vadītāja vai galvenā grāmatveža datus, darbības sfēru, e-pasta adresi vai Interneta lappusi. Informācijas rediģēšana ir pieejama tikai reģistrētiem lietotājiem [16].

Šim uzņēmumu katalogam arī ir gan savas priekšrocības, gan trūkumi, salīdzinot ar 118 katalogu. Par vienu no labajām īpašībām var uzskaitīt lietotāja iespēju mainīt informāciju par savu uzņēmumu. Galvenais trūkums ir nespēja meklēt uzņēmumu pēc vairākiem parametriem. Tas nozīmē, ka apmeklētājs nevar uzreiz meklēt informāciju, zinot, piemēram, uzņēmuma telefona numuru un adresi.

	118 katalogs	Mapquest	Весь Саратов
Meklēšana pēc nosaukuma	☑	☑	☑
Meklēšana pēc adreses	☑	☑	☑
Meklēšana pēc vairākiem parametriem	☑	☑	☐
Meklēšana pēc tālruņa numura	☑	☐	☑
Meklēšana pēc darbības nozares	☑	☐	☑
Meklēšana pēc vadītāja vārda/uzvārda	☐	☐	☑
Meklēšana pēc grāmatveža vārda/uzvārda	☐	☐	☑
Meklēšana pēc pilsētas	☑	☑	☐
Meklēšana pēc rajona	☑	☐	☐
Meklēšana pēc pasta indeksa	☐	☑	☐
Meklēšana pēc e-pasta adreses	☐	☐	☑
Meklēšana pēc Interneta lappuses	☐	☐	☑
Tuvāko uzņēmumu meklēšana	☐	☑	☐
Sava uzņēmuma informācijas rediģēšana	☐	☐	☑
Adrešu meklēšana kartē	☑	☑	☑
Uzņēmumu meklēšana kartē	☑	☐	☑
Kustība pa kartēm	☑	☑	☐
Robežu noteikšana meklēšanai kartē	☑	☑	☐
Attāluma mērītājs	☑	☐	☐
Punktu saglabāšana kartē	☐	☑	☐
Maršruta veidošana	☐	☑	☐

Tabula 12. Uzņēmumu katalogu salīdzināšana

Tabulā 12. ir apkopoti salīdzināšanas rezultāti. Katra rindiņa atbilst vienai funkcijai, kolonna - vienam uzņēmumu katalogam Internetā. ☑ - nozīmē, ka dotajam katalogam ir pieejama sekojoša funkcija, pretējā gadījumā: ☐ - funkcija nav pieejama. Var redzēt, ka kopējais funkciju skaits 1188 sistēmai ir lielāks. Sistēmai Mapquest ir ļoti attīstītas funkcijas, kas saistās ar kartēm. Saratovas kataloga galvenā īpašība ir meklēšanas parametru saraksts, jo viņš būtiski atšķiras no tiem pašiem sarakstiem 1188 un Mapquest katalogā.

Maršrutu attēlojuma interaktīvajās kartēs iespējas realizācija ir diezgan grūts uzdevums no datu bāzes viedokļa. Ir svarīgi, lai lietotājs varētu ne tikai redzēt maršrutus vienā zīmējumā vai shēmā, bet arī izvēlēties tikai vienu konkrētu maršrutu, kā arī apskatīt izvēlētā maršruta pieturu un laiku sarakstus. Pašlaik šī iespēja Latvijas Internetā ir pieejama tikai Rīgas Satiksmes mājas lapā – www.rigassatiksme.lv. Apskatīsim tās realizācijas priekšrocības un trūkumus, salīdzinot to ar līdzīgu igauņu portālu – Tallinna uhistranspordi kaart (portāla izskatu sk. Zīmējumā 18.).

Zīmējums 18. Tallinas maršrutu interaktīvā karte

Aplūkosim galvenās portālu funkcijas: maršrutu, pieturu un laiku attēlošanu. Tallinas portālā ir pieejams pilsētas maršrutu saraksts, kas ietver autobusus, trolejbusus un tramvajus. Var izvēlēties vienu vai vairākus maršrutus vienlaicīgi un apskatīt līnijas, kas attēlo maršrutus pilsētas interaktīvajā kartē. Pieturas ir apzīmētas ar aplīšiem. Uzbraucot ar kursoru uz līnijas, tiek attēloti maršruta numurs un nosaukums, uzbraucot ar kursoru uz pieturas apzīmējuma, tiek attēlots pieturas nosaukums. Ieslēdzot ‘Naita peatuste nimesid’ režīmu labajā apakšējā stūrī, zīmējumā parādās visu maršruta pieturu nosaukumi. Atsevišķs pieturu saraksts maršrutam nav pieejams.

Uzklikšķinot ar peli uz pieturas, atvēras logs ar kartes fragmentu precīzākā līmenī, kas attēlo punkta apkārtni, bet, nospiežot vēlreiz, atvēras HTML logs ar atiešanas laikiem [17].

Šis Interneta resurss ir ļoti līdzīgs 1188 un Rīgas Satiksmes projektam, tomēr pastāv dažādas atšķirības:

- ✓ 1188 sistēmā ir pieejami atsevišķi autobusu, trolejbusu un tramvaju saraksti;
- ✓ 1188 sistēmā papildus maršruti parādās zem galvenā maršruta pēc šī maršruta izvēles;
- ✓ 1188 sistēmā var apskatīt izvēlētā maršruta pieturu sarakstu;
- ✓ 1188 sistēmā laika saraksts parādās lapas galvenajā logā;
- ✓ 1188 sistēmā nevar redzēt visu pieturu nosaukumus kartē;
- ✓ 1188 sistēmā ir pieejami 6 karšu līmeņi, kurus var pārslēgt ar pogām;

Tabulā 13. ir doti 1188 un Tallinas karšu salīdzināšanas rezultāti.

	1188 karte	Tallinas karte
Visu maršrutu saraksts	☑	☑
Viena transportlīdzekļa veidu maršrutu saraksts	☑	☐
Papildus maršrutu saraksts zem galvenā maršruta	☑	☐
Vairāku maršrutu izvēlēšanās vienlaicīgi	☑	☑
Izvēlētā maršruta pieturu saraksts	☑	☐
Laika saraksts Flash formātā	☑	☐
Laika saraksts HTML formātā	☐	☑
Pieturu nosaukumu attēlošana kartē	☐	☑
Pieturas apkārtnes kartes fragments	☐	☑
Vairāki karšu līmeņi	☑	☐

Tabula 13. Maršrutu karšu salīdzināšana

Katra tabulas rindiņa atbilst vienai funkcijai, kolonna- vienai sistēmai. ☑ - nozīmē, ka dotajai sistēmai pieejama sekojoša funkcija, pretējā gadījumā: ☐ - funkcija nav pieejama.

No salīdzināšanas rezultātiem var secināt, ka 1188 sistēmas iespēju saraksts nav pilns. Ir iespējas, kas var būtiski atvieglot meklēšanu, darbu ar katalogu un interaktīvajām kartēm. Jau tagad 1188 strādā pie punktu saglabāšanas kartē un visīsākā maršruta veidošanas iespēju pievienošanas.

Nobeigums

Darbā sniegts 1188 sistēmas un tās pakalpojumu raksturojums. Ir aprakstītas tādas iespējas, kā uzziņu saņemšana pa tālruni, 1188 privātpersonu un uzņēmumu tālruņu katalogi, 1188 mobilā uzziņa un citi. Detalizētāk apskatīts 1188 uzņēmumu katalogs Internetā, tā vēsture un iespējas.

Nākošā daļa ir veltīta uzņēmumu kataloga datu bāzes struktūras aprakstīšanai, kas satur informāciju par visiem Latvijā reģistrētiem uzņēmumiem, sabiedriskām organizācijām, ārvalstu uzņēmumu pārstāvniecībām kā arī ģeogrāfiskajiem objektiem. Ir apskatītas galvenās datu bāzes tabulas, tai skaitā TAB_CLIENTS- uzņēmumu tabula un TAB_ADDRESS- adrešu tabula, un datu bāzes automātiskā atjaunošana, tās trūkumi. Liela uzmanība tika pievērsta datu bāzes optimizācijas jēdzienam un ātruma palielināšanas uzdevuma iespējamiem risinājumiem.

Darba gaitā tika veikta 1188 uzņēmumu kataloga datu bāzes pārveidošana, kas sastāv no diviem etapiem.

1. Datu atjaunošanas optimizācija. Šī etapa laikā tika izstudēti optimizācijas paņēmieni, kas ļauj vienkāršot un paātrināt datu bāzes atjaunošanas procesu, kā arī aprakstīta un realizēta izvēlēta metode.
2. Jauno iespēju realizācija, kas iekļauj izmaiņas datu bāzē un datu struktūrā. Šī pārveidošana ir nepieciešama jauno funkciju pievienošanai: iespēja radīt katalogā sameklēto uzņēmumu atrašanās vietu Latvijas kartē, kā arī meklēt kartes fragmentu pēc uzņēmumu nosaukumiem un Rīgas pilsētas un piepilsētas transportu maršrutu attēlojums 1188 interaktīvajā kartē.

Datu bāzes pārveidošana tika pabeigta. Izdarītie soļi ne tikai ļoti būtiski samazināja atjaunošanas izpildīšanas laiku, bet arī padarīja šo procesu ērtāku. Parādījās iespēja izvēlēties laiku, kas labāk piemērots datu bāzes atjaunošanai un apmeklētāju skaits nav liels. Samazinājās kļūdas veidošanas risks, kā arī pievienota automātiska veco kļūdu atrašana un labošana.

Jau tagad uzņēmumu katalogā Internetā var apskatīt dažu uzņēmumu atrašanās vietas kartē un Rīgas Satiksmes mājas lapā ir pieejamas 1188 interaktīvās kartes ar iespēju apskatīt maršrutu attēlojumus kartē, to pieturu un laiku sarakstus. Darbā ir dots papildināta 1188 kataloga salīdzinājums ar citiem līdzīgiem ārvalstu katalogiem Internetā.

Tomēr palika tālākas datu bāzes pārveidošanas perspektīvas, visgalvenākā no kuriem ir maršrutu nodaļas papildināšana ar optimālā maršruta atrašanas iespēju.

Literatūras saraksts

1. Uzziņu dienests 118 maina numuru uz 1188 [tiešsaite]. [atsauce 11.02.2006.]. Pieejams Internetā: http://www.lattelekom.lv/ltk/content/?id=12560
2. 118 uzziņu dienests [tiešsaiste]. [atsauce 13.01.2005.]. Pieejams Internetā: http://ads.118.lv/pakalpojumi/?page=118#118
3. Uzziņu dienests 118 [tiešsaiste]. [atsauce 13.01.2005.]. Pieejams Internetā: http://www.lattelekom.lv/ltk/content/?cat=280
4. 1188 tālruņu katalogi [tiešsaiste]. [atsauce 20.01.2005.]. Pieejams Internetā: http://www.interinfo.lv/356
5. E-reģistrs [tiešsaiste]. [atsauce 21.01.2005.]. Pieejams Internetā: http://www.interinfo.lv/403
6. Pieejas kanāli [tiešsaiste]. [atsauce 21.01.2005.]. Pieejams Internetā: http://www.interinfo.lv/386
7. Jauns Uzziņu dienesta 118 pakalpojums – 118 mobilā uzziņa [tiešsaiste]. 2001.gada decembris [atsauce 2.02.2005.]. Pieejams Internetā: http://www.lattelekom.lv/ltk/content/?cat=2057&lang=lat
8. Sybase Adaptive Server [tiešsaiste]. [atsauce 25.03.2006.]. Pieejams Internetā: http://www.docflow.ru/full_news.asp?param=29798
9. Sybase Adaptive Server Enterprise [tiešsaiste]. [atsauce 30.03.2006.]. Pieejams Internetā: http://www.sybase.lv/adaptiveserverenterprise.html
10. Androutsopoulos I., Ritchie G.D., Thanisch P. Natural Language Interfaces to Databases - An Introduction: Natural Language Engineering, 1995. – 29 lpp.
11. Д. Шаша, Ф. Бонне. Оптимизация баз данных. Принципы, практика, решение проблем: КУДИЦ-ОБРАЗ,2004. – 13 lpp.
12. Serge Abiteboul, Richard Hull, Victor Vianu. Foundations of Databases: Wesley – Addison, 1995. – 67 lpp.
13. С.Кузнецов. Методы оптимизации выполнения запросов в реляционных СУБД [tiešsaiste]. [atsauce 15.04.2005.]. Pieejams Internetā: http://www.citforum.ru/database/articles/art_26.shtml
14. SIA Rīgas satiksme [tiešsaite]. [atsauce 7.04.2006]. Pieejams Internetā: http://www.rigassatiksme.lv/
15. Mapquest [tiešsaite]. [atsauce 14.02.2005.]. Pieejams Internetā: http://www.mapquest.com
16. Весь Саратов [tiešsaite]. [atsauce 15.02.2005]. Pieejams Internetā: http://www.sarbc.ru/saratov/
17. Tallinna uhistranspordi kaart [tešsaite]. [atsauce 17.04.2006]. Pieejams Internetā: http://veeb.tallinn.ee/transport/pikasa/scripts.php

Apliecinājums

Ar šo es apliecinu, ka šodien iesniegto bakalaura darbu es esmu veicis pašrocīgi un esmu izmantojis tikai tajā norādītos palīglīdzekļus.

Rīgā, Paraksts:

Pielikums

1.TAB_LTKIMPORT update

```
ALTER trigger
admin.update_ltkimport after update order 2 on
admin.TAB_LTKIMPORT
referencing old as old_imp new as new_imp
for each row
begin
  declare cn integer;
  declare i integer;
  declare ec varchar(64);
  declare k integer;
  //check quantity of clients with ltk_code
  select count(*) into cn from admin.tab_ltkfullbase
    where ltk_code = new_imp.ltk_code;
  set cn=ifnull(cn,0,cn);
  set i=ifnull(new_imp.ltk_code,0,new_imp.ltk_code);
  //check type (what to do)
  if new_imp.importok = 0 then
    if i > 0 then
      if(new_imp.importtype = 12) then
        // delete type
        update admin.tab_ltkimport set importok = 0
          where id_ltkimport = new_imp.id_ltkimport;
        update admin.tab_ltkfullbase set ltkimport = new_imp.id_ltkimport,
          lastdate = new_imp.loaddate,status = 0
          where ltk_code = new_imp.ltk_code
      else
        if(new_imp.importtype = 11) or(new_imp.importtype = 13) then
          // Insert new or update type (1 or 3)
          set ec=null;
          set k=ifnull(new_imp.name,0,1);
          if k = 1 then
            set k=length(new_imp.name)
          end if;
          if k > 0 then
            if cn = 0 then
              insert into admin.tab_ltkfullbase(ltk_code,
                name,fullheading,address,city,region,zip,
                phone,lastdate,newheadings,status,lur) values(
                new_imp.ltk_code,new_imp.name,
                new_imp.fullheading,new_imp.address,
                new_imp.city,new_imp.region,new_imp.zip,
                new_imp.phone,new_imp.loaddate,
                new_imp.newheadings,1,new_imp.lur)
            else
              update admin.tab_ltkfullbase set
                ltk_date = new_imp.ltk_code,
                name = new_imp.name,
                fullheading = new_imp.fullheading,
                city = new_imp.city,
                region = new_imp.region,
                zip = new_imp.zip,
                phone = new_imp.phone,
                lastdate = new_imp.loaddate,
                newheadings = new_imp.newheadings,
                status = 2,
                lur = new_imp.lur
                where ltk_code = new_imp.ltk_code;
            end if;
```

```
          else
            set ec='Company name not exist';
            update admin.tab_ltkimport set
              importok = 3,
              importerror = ec
              where id_ltkimport = new_imp.id_ltkimport
          end if
        end if
      end if
    else
      set ec='LTK code is 0 or not recognized';
      update admin.tab_ltkimport set
        importok = 2,
        importerror = ec
        where id_ltkimport = new_imp.id_ltkimport
    end if
  end if
end
```

2.TAB_LTKFULLBASE insert

```
ALTER trigger
admin.ins_ltkfullbase after insert order 1 on
admin.TAB_LTKFULLBASE
referencing new as new_fb
for each row
begin
  //insert clients into tab_ltkclients
  if(new_fb.status = 1) or(new_fb.status = 2) then
    insert into admin.tab_ltkclients(ltk_code,
      name,fullheading,address,city,region,zip,
      phone,lastdate,newheadings,lur) values(
      new_fb.ltk_code,new_fb.name,new_fb.fullheading,
      new_fb.address,new_fb.city,new_fb.region,new_fb.zip,
      new_fb.phone,new_fb.lastdate,
      new_fb.newheadings,new_fb.lur);
    //check client in database
    call internet.ltkcheckclient(new_fb.ltk_code)
  else
  //disable client in database
    update admin.tab_clients set status = 8
      where ltk_code = new_fb.ltk_code
  end if
end
```

3.TAB_LTKFULLBASE update

```
ALTER trigger
admin.upd_fullbase after update order 2 on
admin.TAB_LTKFULLBASE
referencing old as old_fb new as new_fb
for each row
begin
  declare cn integer;
  //check type(what to do)
  if(new_fb.status = 1) or(new_fb.status = 2) then
  //insert or update, count clients with same ltk_code
    select count(*) into cn from admin.tab_ltkclients
      where ltk_code = new_fb.ltk_code;
    if cn = 0 then
      insert into admin.tab_ltkclients(ltk_code,
        name,fullheading,address,city,region,zip,
        phone,lastdate,newheadings,lur) values(
        new_fb.ltk_code,new_fb.name,new_fb.fullheading,
        new_fb.address,new_fb.city,new_fb.region,new_fb.zip,
        new_fb.phone,new_fb.lastdate,new_fb.newheadings,new_fb.lur)
    else
      update admin.tab_ltkclients set
        name = new_fb.name,
        fullheading = new_fb.fullheading,
        address = new_fb.address,
        city = new_fb.city,
        region = new_fb.region,
        zip = new_fb.zip,
        phone = new_fb.phone,
        lastdate = new_fb.lastdate,
        newheadings = new_fb.newheadings,
        lur = new_fb.lur
        where ltk_code = new_fb.ltk_code
    end if;
    //check client in TAB_CLIENTS
    call internet.ltkcheckclient(new_fb.ltk_code)
  else
    //disable client
    delete from admin.tab_ltkclients
      where ltk_code = new_fb.ltk_code;
    update admin.tab_clients set status = 8
      where ltk_code = new_fb.ltk_code
  end if
end
```

4.LTKCHECKCLIENT procedure

```
ALTER procedure
internet.ltkcheckclient(in ltk_cd numeric(7))
begin
  declare i integer;
  //Client status into LTK database
  declare ltkstat integer;
  // order for verifying RD dataabse
  declare rdcommand integer;
  //How many ltk_code exists into RD
  declare rdlc_cnt integer;
  declare rdsc numeric(9,2);
  declare idc numeric(9);
  declare idcs numeric(9);
  declare clientsphone numeric(7);
  declare ltkphone numeric(7);
  declare clientsnewheadings numeric(9);
  declare ltknewheadings numeric(9);
  declare clientsstatus integer;
  declare clientsname varchar(256);
  declare ltkname varchar(256);
  declare clientsaddress varchar(256);
  declare ltkaddress varchar(256);
  declare clientsaddr_city varchar(64);
  declare ltkcity varchar(64);
  declare clientsaddr_region varchar(64);
  declare ltkregion varchar(64);
  declare clientszip numeric(4);
  declare ltkzip numeric(4);
  //address coordinates
  declare xspec numeric(9);
  declare yspec numeric(9);
  //address special form
  declare specc varchar(64);
  declare speccity varchar(64);
  declare idspec numeric(9);
  set specc='';
  set i =ifnull(ltk_cd,0,ltk_cd);
  if i > 0 then
    set rdcommand=999
    // Continue checking
  else
    print 'LTK_CODE is null or 0';
    set rdcommand=0
  end if;
  if rdcommand = 999 then
    // check client status in LTK database
    // 0 - client exist into LTK, but deleted
    // 1 - client exist into database (one time inserted - without statistics)
    // 2 - client exist into LTK database and has more than one time updated
    // -1 - client not exist into LTK database
    select status into ltkstat
      from admin.tab_ltkfullbase
      where ltk_code = ltk_cd;
    set ltkstat=ifnull(ltkstat,-1,ltkstat);
    case ltkstat
    when-1 then
      print 'Client %1! not exist into LTK database',ltk_cd;
      set rdcommand=1
      // Set client deleted into RD
    when 0 then
      print 'Client %1! is deleted from LTK database',ltk_cd;
```

```
    set rdcommand=1
    // Set client deleted into RD
  when 1 then
    set rdcommand=2
    // Check client into RD
  when 2 then
    set rdcommand=2
    // Check client into RD
  else
    set rdcommand=1
    // Set client deleted into RD
  end case
end if;
if rdcommand = 1 then
  // Set client as deleted into RD database if order is 1
  update admin.tab_clients set
    status = 8
    where ltk_code = ltk_cd;
  set rdcommand=0
  // Finish
end if;
if rdcommand = 2 then
  // Check client into RD
  // Check for LTK_CODE duplicates into database
  select count(ltk_code) into rdlc_cnt
    from admin.tab_clients
    where ltk_code = ltk_cd and
    (tab_clients.status <> 8 or tab_clients.status is null);
  set rdlc_cnt=ifnull(rdlc_cnt,0,rdlc_cnt);
  case rdlc_cnt
  when 0 then
    // Client not exists into RD database - should be created
    print 'Create client %1! into RD database',ltk_cd;
    set rdcommand=3
    // Create client into rd database
  when 1 then
    // Client exists into RD database - compare RD and LTK
    print 'Compare client %1! into RD and LTK databases',ltk_cd;
    set rdcommand=4 // Compare client into LTK and RD databases
    // LTK_CODE duplicates... duplicate checking!
  else
    print 'Client %1! duplcates (%2!) into RD database',ltk_cd,rdlc_cnt;
    set rdcommand=5
    // Destroy duplicates into RD
  end case
end if;
if rdcommand = 3 then
  // Create client into rd database or chanege status
  select count(ltk_code) into rdlc_cnt
    from admin.tab_clients where
    ltk_code = ltk_cd and tab_clients.status = 8;
  if rdlc_cnt > 0 then
    select min(tab_clients.idr_clients) into idc
      from admin.tab_clients
      where tab_clients.ltk_code = ltk_cd;
    //check for address change
    if(select address from admin.tab_clients
       where id_clients=idc) <>
      (select address from admin.tab_ltkclients
       where ltk_code=ltk_cd) then
      select address into ltkaddress from admin.tab_ltkclients
        where ltk_code=ltk_cd;
      update tab_clients set tab_clients.specaddr =
       ('.' ||trim(internet.normalizetxt(replace(ltkaddress,' iela',' ')))
```

```
        || '.');
      set specc='.';
    end if;
    update admin.tab_clients set
      tab_clients.iiphone = tab_ltkclients.phone,
      tab_clients.newheadings = tab_ltkclients.newheadings,
      tab_clients.status = 6,
      tab_clients.name = tab_ltkclients.name,
      tab_clients.address = tab_ltkclients.address,
      tab_clients.addr_city = tab_ltkclients.city,
      tab_clients.addr_region = tab_ltkclients.region,
      tab_clients.zip = tab_ltkclients.zip,
      tab_clients.ltk_ur =
        ifnull(tab_ltkclients.lur,null,tab_ltkclients.lur),
      from admin.tab_clients,admin.tab_ltkclients
      where tab_clients.ltk_code = tab_ltkclients.ltk_code and
        tab_ltkclients.ltk_code = ltk_cd and
        tab_clients.idr_clients = idc
  else
    insert into admin.tab_clients(ltk_code,name,address,
      addr_city,zip,iiphone,newheadings,status)
      select ltk_code,name,address,city,zip,phone,newheadings,6
        from admin.tab_ltkclients where ltk_code = ltk_cd
  end if;
  set rdcommand=0
  //Finish
end if;
if rdcommand = 4 then
  // Compare client into LTK and RD databases
  select tab_clients.iiphone,tab_clients.newheadings,
    tab_clients.status,tab_clients.name,
    tab_clients.address,tab_clients.addr_city,
    tab_clients.addr_region, tab_clients.zip,
    into clientsphone, clientsnewheadings,
    clientsstatus,clientsname,clientsaddress,
    clientsaddr_city,clientsaddr_region,
    clientszip
    from admin.tab_clients
    where tab_clients.ltk_code = ltk_cd and status <> 8;
  select tab_ltkclients.phone,tab_ltkclients.newheadings,
    tab_ltkclients.name,tab_ltkclients.address,
    tab_ltkclients.city,tab_ltkclients.region,
    tab_ltkclients.zip
    into ltkphone,ltknewheadings,ltkname,
    ltkaddress,ltkcity,ltkregion,ltkzip
    from admin.tab_ltkclients
    where tab_ltkclients.ltk_code = ltk_cd;
  //check fields for null
  set clientsphone=
    ifnull(clientsphone,0,clientsphone);
  set clientsnewheadings=
    ifnull(clientsnewheadings,0,clientsnewheadings);
  set clientsstatus=
    ifnull(clientsstatus,0,clientsstatus);
  set clientsname=
    ifnull(clientsname,'',clientsname);
  set clientsaddress=
    ifnull(clientsaddress,'',clientsaddress);
  set clientsaddr_city=
    ifnull(clientsaddr_city,'',clientsaddr_city);
  set clientsaddr_region=
    ifnull(clientsaddr_region,'',clientsaddr_region);
  set clientszip=
    ifnull(clientszip,0,clientszip);
```

```
  set ltkphone=
    ifnull(ltkphone,0,ltkphone);
  set ltknewheadings=
    ifnull(ltknewheadings,0,ltknewheadings);
  set ltkname=
    ifnull(ltkname,'',ltkname);
  set ltkaddress=
    ifnull(ltkaddress,'',ltkaddress);
  set ltkcity=
    ifnull(ltkcity,'',ltkcity);
  set ltkregion=
    ifnull(ltkregion,'',ltkregion);
  set ltkzip=
    ifnull(ltkzip,0,ltkzip);
 //check for address change
 if (clientsaddress <> ltkaddress) then
   update tab_clients set tab_clients.specaddr =
    ('.' ||trim(internet.normalizetxt(replace(ltkaddress,' iela',' ')))
      || '.');
    set specc='.';
 end if;
//check for fields change
  if(clientsphone <> ltkphone)
    or(clientsnewheadings <> ltknewheadings)
    or(clientsstatus <> 6)
    or(clientsname <> ltkname)
    or(clientsaddress <> ltkaddress)
    or(clientsaddr_city <> ltkcity)
    or(clientsaddr_region <> ltkregion)
    or(clientszip <> ltkzip)
    then
      //if something changed
      update admin.tab_clients set
        tab_clients.iiphone = tab_ltkclients.phone,
        tab_clients.newheadings = tab_ltkclients.newheadings,
        tab_clients.status = 6,
        tab_clients.name = tab_ltkclients.name,
        tab_clients.address = tab_ltkclients.address,
        tab_clients.addr_city = tab_ltkclients.city,
        tab_clients.addr_region = tab_ltkclients.region,
        tab_clients.zip = tab_ltkclients.zip,
        from admin.tab_clients,admin.tab_ltkclients
        where tab_clients.ltk_code = tab_ltkclients.ltk_code and
        tab_ltkclients.ltk_code = ltk_cd
  end if;
  set rdcommand=0 //Finish
end if;
if rdcommand = 5 then
  // Destroy duplicates into RD
  // Get info for contracts
  select count(*) into rdsc
    from tab_contract,tab_clients
    where tab_contract_clients=tab_clients.id_clients and
      tab_clients.ltk_code=ltk_cd;
  // If scale is > 0 then this client is first else client with min id
  if rdsc > 0 then
    select top 1 tab_clients.id_clients into idc
      from admin.tab_clients
      where (tab_clients.status <> 8 or tab_clients.status is null) and
      (select count(*) from tab_contract
         where clients =tab_clients.id_clients) > 0 and
       tab_clients.ltk_code=ltk_cd
  else
    select min(id_clients) into idc
```

```
      where (tab_clients.status <> 8 or tab_clients.status is null) and
      tab_clients.ltk_code=ltk_cd
  end if;
  // All other clients should be disabled
  update admin.tab_clients set
    status = 8,
    ltk_code = null
    where tab_clients.ltk_code = ltk_cd and tab_clients.idr_clients <> idc;
  // Update active client info!
  //change for address change
  if(select address from admin.tab_clients
     where id_clients=idc) <>
    (select address from admin.tab_ltkclients
     where ltk_code=ltk_cd) then
    select address into ltkaddress from admin.tab_ltkclients
      where ltk_code=ltk_cd;
    update tab_clients set tab_clients.specaddr =
     ('.' ||trim(internet.normalizetxt(replace(ltkaddress,' iela',' ')))
      || '.');
    //if address was changed edit specc variable
    set specc='.';
  end if;
  update admin.tab_clients set
    tab_clients.iiphone = tab_ltkclients.phone,
    tab_clients.newheadings = tab_ltkclients.newheadings,
    tab_clients.status = 6,
    tab_clients.name = tab_ltkclients.name,
    tab_clients.address = tab_ltkclients.address,
    tab_clients.addr_city = tab_ltkclients.city,
    tab_clients.addr_region = tab_ltkclients.region,
    tab_clients.zip = tab_ltkclients.zip,
    from admin.tab_clients,admin.tab_ltkclients
    where tab_clients.ltk_code = tab_ltkclients.ltk_code and
      tab_ltkclients.ltk_code = ltk_cd and
      tab_clients.idr_clients = idc;
  set rdcommand=0 //finish;
end if;
//id address were updated
if (specc <> '') then
  select specaddr,addr_city into specc,speccity
    from admin.tab_clients
    where ltk_code = ltk_cd and status = 6;
  //check first word
  if(specc not like '.maza%') and(specc not like '.Britu%')
    and(specc not like '.Mazais%') and(specc not like '.Lielais%') and
    (specc not like '.Jauna%') and(specc not like '.Liela%') and
    (specc not like '.Gunara%') and(specc not like '.Veca%') then
      //first word is good
      if(specc like '%birznieka%') then
        select top 1 id_address,x,y into idspec,xspec,yspec
          from tab_address
          where locate(specc,'.e' || easyname) > 0 and
            city = new_imp.city
      else
        select top 1 id_address,x,y into idspec,xspec,yspec
          from tab_address
          where locate(specc,easyname) > 0 and
            city = new_imp.city
      end if
  else
    //first word is bad
    select top 1 id_address,x,y into idspec,xspec,yspec
      from tab_address
      where locate(specc,easyname) = 1 and
```

```
          city = new_imp.city
    end if;
    update admin.tab_clients set
      addrid = idspec,
      x = xspec,
      y = yspec
      where ltk_code = new_imp.ltk_cd and status=6;
  return
end
```

5.NORMALIZETXT function

```
ALTER function
internet.normalizetxt(in @tx varchar(256))
returns varchar(256)
begin
  declare @ntwrd varchar(256);
  declare @nt varchar(256);
  declare @novalidchars varchar(64);
  declare @ignorechars varchar(64);
  declare @qchars varchar(64);
  declare @ntch char;
  declare @ch char;
  declare @ch2 char;
  declare @i integer;
  declare @k integer;
  declare @kk integer;
  declare @cl integer;
  declare @isnum integer;
  declare @lnm integer;
  declare @lc integer;
  declare @k1 integer;
  declare @k2 integer;
  declare @lchars varchar(96);
  declare @echars varchar(96);
  //read no valid, Latvian and English characters
  select nt_novalidchars,nt_ignorechars,nt_char,latchars,engchars
    into @novalidchars,@ignorechars,@ntch,@lchars,@echars
    from admin.tab_idefault;
  //check no valid symbols
  set @qchars=isnull(@novalidchars,'0');
  set @i=1;
  if @qchars = '0' then
    set @i=0
  end if;
  // if value is null, add bad symbols
  if @i = 0 then
    set @novalidchars=
      '`~!@#$%^&*()_-+=' || "char"(39) || '{[}]|\\:;"<,>. ?/|\\';
    update admin.tab_idefault set
      tab_idefault.nt_novalidchars = @novalidchars
  end if;
  set @novalidchars=@novalidchars || ' ';
  //check ignored symbols
  set @qchars=isnull(@ignorechars,'0');
  set @i=1;
  if @qchars = '0' then
    set @i=0
  end if;
  // if there is not ignored symbols
  if @i = 0 then
    set @ignorechars='`~"' || "char"(39);
    update admin.tab_idefault set
      tab_idefault.nt_ignorechars = @ignorechars
  end if;
  set @qchars=isnull(@ntch,'0');
  set @i=1;
  if @qchars = '0' then
    set @i=0
  end if;
  if @i = 0 then
    set @ntch='.';
    update admin.tab_idefault set
```

```
      tab_idefault.nt_char = rtrim(@ntch)
  end if;
  set @lnm=char_length(@tx);
  set @lnm=isnull(@lnm,0)+1;
  set @i=1;
  set @nt='';
  set @ntwrd='';
  set @lc=1;
  set @isnum=0;
  while @i <= @lnm+1 loop
    if @i = @lnm then
      set @ch=' '
    else
      set @ch=substring(@tx,@i,1)
    end if;
    //find symbol's position in ignorechars
    set @k1=charindex(@ch,@ignorechars);
    set @k1=isnull(@k1,0);
    // not in ignorechars
    if @k1 = 0 then
      //find symbol's position in novalidchars
      set @k2=charindex(@ch,@novalidchars);
      set @k2=isnull(@k2,0);
      if @k2 = 0 then
        //find symbol's position in Latvian chars
        set @kk=charindex(@ch,@lchars);
        set @kk=isnull(@kk,0);
        if @kk > 0 then
          //replace with English symbol
          set @ch2=substring(@echars,@kk,1)
        else
          set @ch2=@ch
        end if;
        set @ntwrd=@ntwrd+@ch2;
        set @lc=0;
        set @k=charindex(@ch,'0123456789');
        if @k > 0 and @isnum < 2 then
          set @isnum=1
        else
          set @isnum=2
        end if
      else
        if @lc = 0 then
          set @ntwrd=ltrim(@ntwrd);
          set @cl=char_length(@ntwrd);
          if @i <> @lnm then
            set @ntwrd=rtrim(@ntwrd)+rtrim(@ntch)
          end if;
          set @isnum=0;
          set @nt=@nt+ltrim(@ntwrd);
          set @ntwrd='';
          set @lc=1
        end if
      end if
    end if;
    set @i=@i+1
  end loop;
  set @nt=ltrim(@nt);
  //Convert characters in a string to lower case, replace '..' with '.'
  set @nt=replace(lower(@nt),'..','.');
  return @nt
end
```

6.ADREASYNAME procedure

```
ALTER procedure internet.adreasyname()
//Procedura, kas aizpilda easyname lauku
begin
  declare err_notfound exception for sqlstate value '02000';
  declare addrf numeric(9);
  declare addrt numeric(9);
  declare ida numeric(9);
  declare nm varchar(64);
  declare nmn varchar(32);
  declare i numeric(9);
  declare k numeric(9);
  declare ch varchar;
  declare qq_adr dynamic scroll cursor for
    select id_address,name,namenum
      from admin.tab_address
      where id_address >= addrf
      order by id_address asc;
  declare s varchar(64);
  declare sub varchar(64);
  select min(id_address) into addrf from admin.tab_address;
  select max(id_address) into addrt from admin.tab_address;
  set i=addrf-1;
  open qq_adr;
  qq_loop: loop
    //nolasa adreshu tabulas rindu
    fetch next qq_adr into ida,nm,nmn;
    if(sqlstate = err_notfound) or(i > 100000) then
      leave qq_loop
    end if;
    set i=i+1;
    if i >= addrf and i <= addrt then
      set s=replace(nm,' iela',' ');
      set s=replace(s,'prosp.','prospekts');
      //apstrada latvieshu alfabeta burtus un pieturas zimes
      set s=trim(internet.normalizetxt (s));
      set k=1;
      set sub='';
      //nolasa pirmo ielas nosaukuma vaardu
      while substring(s,k,1) <> '.' and k < length(s) loop
        set sub=sub || substring(s,k,1);
        set k=k+1
      end loop;
      //vai pirmais vards ir ‘slikts’
      if(sub not like 'maza') and(sub not like 'Britu') and
        (sub not like 'Mazais') and(sub not like 'Lielais') and
        (sub not like 'Jauna') and(sub not like 'Liela') and
        (sub not like 'Gunara') and(sub not like 'Veca') and
        (k < length(s)) then
        set sub=substring(s,k+1);
        set ch=substring(sub,length(sub),1);
        if ch <> '.' then
          set sub=sub || '.'
        end if;
        if sub = 'mstislava.keldisa.' then set sub='keldisa.'
        end if;
        //vai palikushais vards ir ‘slikts’
        if(sub = 'prospekts.') or(sub = 'laukums.') or
          (sub like '%linija.') or(sub = 'soseja.') or
          (sub = 'aleja.') or (sub = 'gatve.') or
          (sub = 'seta.') or(sub = 'cels.') or
          (sub = 'skvers.') or(sub = 'bulvaris.') or
```

```
          (sub = 'dambis.') or(sub = 'magistrale.') or
          (sub = 'krastmala.') or(sub = 'gate.') or
          (sub = 'ielas.') or (sub = 'skersiela.') then
          set sub=s
        end if;
        set ch=substring(sub,length(sub),1);
        if ch <> '.' then
          set sub=sub || '.'
        end if;
        //ja pirmo vardu var dzest
        set sub='.' || sub || trim(normalizetxt(nmn)) || '.';
        update admin.tab_address set easyname = sub where id_address = ida
      else
        //pirmo vardu nevar dzest
        set ch=substring(s,length(s),1);
        if ch <> '.' then
          set s=s || '.'
        end if;
        set s='.' || s || trim(normalizetxt(nmn)) || '.';
        update admin.tab_address set
          easyname = s
          where id_address = ida
      end if
    end if;
    if i > addrt then
      leave qq_loop
    end if
  end loop qq_loop;
  close qq_adr;
  print 'Adreasyname done';
  return
end
```

7.ADDRESSUPDATE procedure

```
ALTER procedure
internet.addressupdate(in addrf numeric(9),in addrt numeric(9))
//Proceduura, kas ieraksta addrid,x,y tabulaa TAB_CLIENTS
begin
  declare err_notfound exception for sqlstate value '02000';
  declare ida numeric(9);
  declare nm varchar(64);
  declare ct varchar(32);
  declare i numeric(9);
  declare acn numeric(9);
  declare ch char;
  declare xx numeric(9);
  declare yy numeric(9);
  declare au integer;
  declare plid numeric(9);
  declare ltkid numeric(15);
  declare skait numeric(9);
  declare c numeric(9);
  declare sub varchar(64);
  //kursors, kas nolasa pa vienu rindinju no TAB_ADDRESS
  declare vw_ltk dynamic scroll cursor for
    select id_address,easyname,city,x,y,id_place
    from admin.tab_address,admin.tab_place
    where admin.tab_address.city = admin.tab_place.place and
    id_address >= addrf
    order by id_address asc;
  //kursors, kas nolasa klientus ar vajadziigo pasta indeksu
  declare qq_ltk dynamic scroll cursor for
    select id_clients
      from #temp_tab,admin.tab_clients
      where zipfr <= admin.tab_clients.zip and
        zipto >= admin.tab_slients.zip and
        (acn > 0 and specaddr like '%' || nm || '%')
      order by id_clients asc;
  set au=0;
  set c=100;
  //cik kopaa jaapstraadaa
  select count(*) into skait
    from admin.tab_address,admin.tab_place
    where admin.tab_address.city = admin.tab_place.place and
      id_address >= addrf;
  if au = 0 then
    set i=addrf-1;
    open vw_ltk;
    vw_loop: loop
      fetch next vw_ltk into ida,nm,ct,xx,yy,plid;
      if(sqlstate = err_notfound) or(i > 100000) then
        leave vw_loop
      end if;
      set i=i+1;
      //apstraadot simts ierakstus, rakstaam kopeejo apstraadaato skaitu
      if i = c then
        print 'Apstraadaajis "%1! / %2!" ',c,skait;
        set c=c+100
      end if;
      if i >= addrf and i <= addrt then
        //cik klientu ar vajadziigo adresi
        select count(*) into acn
          from admin.tab_clients
          where (specaddr like '%' || nm || '%') and city = ct;
        if(acn > 0) then
```

```
          select id_zipplace,zipfr,zipto,place
            into #temp_tab
            from admin.tab_zipplace
            where place = plid;
          open qq_ltk;
          qq_loop: loop
            fetch next qq_ltk into ltkid;
            if(sqlstate = err_notfound) then
              leave qq_loop
            end if;
            select specaddr into sub
              from admin.tab_clients
                where id_clients = ltkid;
            if(sub not like '.maza%') and(sub not like '.Britu%')and
              (sub not like '.Mazais%') and(sub not like '.Lielais%') and
              (sub not like '.Jauna%') and(sub not like '.Liela%') and
              (sub not like '.Gunara%') and(sub not like '.Veca%') then
              if(sub like '%birznieka%') then
                set nm='.e' || nm
              else
        //ja pirmais vaards nav 'slikts' varam pa kreisi papildinaat ar jebko
                set nm='%' || nm
              end if
            end if;
            update admin.tab_clients set
              addrid = ida,
              x = xx,
              y = yy
              where id_clients = ltkid and specaddr like nm || '%'
          end loop qq_loop;
          close qq_ltk;
          drop table #temp_tab
        end if
      end if;
      if i > addrt then
        leave vw_loop
      end if
    end loop vw_loop;
    close vw_ltk;
    print 'ADDRESSUPDATE visus! ierakstus ir apstradajis!'
  end if;
  return
end
```

8.GETMARSH procedure

```
ALTER procedure internet.Anny_getmarsh()
//Procedura kas no RS tabulaam aizpilda TAB_MARSHRUTS, TAB_MARSHCOORD,
//TAB_MARSHTIMESTAMPS, TAB_MARSHTIME
begin
  declare err_notfound exception for sqlstate value '02000';
  declare virziens integer;
  declare pretvirziens integer;
  declare ida integer; //marshruta id RS tabulaa
  declare minmas integer;
  declare marshname varchar(255);
  declare marshnumber varchar(50);
  declare marshtypes varchar(20);
  declare marshattribute varchar(20);
  declare marshid numeric(9); //marshruta id TAB_MARSHRUTS tabulaa
  declare marshcoordid numeric(9);
  declare xx real;
  declare yy real;
  declare namesak varchar(255); //pirmaas pieturas nosaukums
  declare dotname varchar(255);
  declare dottype integer;
  declare dotplace integer;
  declare dotnum varchar(10);
  declare dotvirziens integer;
  declare dotid integer;
  declare atkarpa integer;
  declare prevname varchar(255);
  declare prevdotnum varchar(10);
  declare timeh integer;
  declare timem integer;
  declare i integer;
  declare checkp integer;
  declare transp integer;
  //kursors kas atlasa marshrutus
  declare vw_cur dynamic scroll cursor for
    select t1.id as tid, (case transportas
                  when 'Autobuss' then '1'
                  when 'Autob_reg' then '2'
                  when 'Tramvajs' then '3'
                  when 'Trolejbuss' then '4' end) as Transport,
         t1.marsrutas, t2.pavadinimas, t2.id as kid, t1.tvarkarastis,
         t3.atributas
    from admin.tTvarkarasciai as t1, admin.tkryptys as t2,
         admin.tTvarkarasciutrukmiuatributai as t3
    where t2.tipas = 'A>B' and t1.id = t2.idtvarkarastis and
          t3.idTvarkarastis =  t1.id
union select idmarsh as tid, (case transportas
                  when 'Autobuss' then '1'
                  when 'Autob_reg' then '2'
                  when 'Tramvajs' then '3'
                  when 'Trolejbuss' then '4' end) as Transport,
         t1.marsrutas||' *', t2.pavadinimas, idkr as kid, t1.tvarkarastis,
         t3.atributas
    from admin.tTvarkarasciai as t1, admin.tkryptys as t2,
         admin.tTvarkarasciutrukmiuatributai as t3, TAB_TEMPMARSH
    where t1.id = t2.idtvarkarastis and
          t3.idTvarkarastis =  t1.id and t1.id=idmarsh and t2.id=idkr;
  //kursors kas atlasa puntus un pieturas
  declare qq_cur dynamic scroll cursor for
    select x, y, pavadinimas, t3.tipas, t3.num, t1.idkryptis, t2.idStotele,
           t2.idAtkarpa
    from admin.tlaikai as t1,admin.tatkarposestoteles as t2
```

```
  left outer join admin.tatkarpos as t3
    on t2.idstotele = t3.id and t3.tipas = 4
  where (t1.idatkarpa = t2.idatkarpa) and ((x is not null) and
        ((t2.nuotolis = 0 or t2.nuotolis is null) and
        (t3.tipas <> 4 or t3.tipas is null) or t3.tipas = 4)) and
        t1.idtvarkarastis = ida and idmasina = minmas and
        t1.idkryptis in(virziens,pretvirziens)
  order by atvyksta asc;
//kursors kas atlasa pieturu laikus
declare time_cur dynamic scroll cursor for
  select truncnum((t1.atvyksta+t1.Trukme*
   (select ifnull(sum(nuotolis),0,sum(nuotolis))
    from admin.tatkarposestoteles
    where idatkarpa = atkarpa and nuotolis <> 0 and id <=
     (select min(t2.id)
      from admin.tLaikai as t1,admin.tAtkarposeStoteles as t2
      where t1.idtvarkarastis = ida and idstotele = dotid and
            t1.idatkarpa=t2.idatkarpa))/(t3.nuotolis+.000001))/60,0),
    round(mod(t1.atvyksta+t1.Trukme*
     (select ifnull(sum(nuotolis),0,sum(nuotolis))
      from admin.tatkarposestoteles
      where idatkarpa = atkarpa and nuotolis <> 0 and id <=
       (select min(t2.id)
        from admin.tLaikai as t1,admin.tAtkarposeStoteles as t2
        where t1.idtvarkarastis = ida and idstotele = dotid and
              t1.idatkarpa = t2.idatkarpa))/(t3.nuotolis+.000001),60),0)
  from admin.tlaikai as t1,admin.tatkarposestoteles as t2,
       admin.tatkarpos as t3
  where t1.idatkarpa = t2.idatkarpa and t1.idatkarpa = t3.id and
        t2.idstotele = dotid and t1.tipas in(0,8) and
        t1.idatkarpa = atkarpa   and
        idtvarkarastis = ida and t1.idkryptis in(virziens,pretvirziens)
  order by atvyksta asc;
//kursors kas atlasa galapunktu atieshanas laikus
declare time2_cur dynamic scroll cursor for
  select truncnum((t1.isvyksta+t1.Trukme*
   (select ifnull(sum(nuotolis),0,sum(nuotolis))
    from admin.tatkarposestoteles
    where idatkarpa = atkarpa and nuotolis <> 0 and id <=
     (select min(t2.id)
      from admin.tLaikai as t1,admin.tAtkarposeStoteles as t2
      where t1.idtvarkarastis = ida and idstotele = dotid and
      t1.idatkarpa = t2.idatkarpa))/(t3.nuotolis+.000000000000000001))/60,0),
    round(mod(t1.isvyksta+t1.Trukme*
     (select ifnull(sum(nuotolis),0,sum(nuotolis))
      from admin.tatkarposestoteles
      where idatkarpa = atkarpa and nuotolis <> 0 and id <=
       (select min(t2.id)
        from admin.tLaikai as t1,admin.tAtkarposeStoteles as t2
        where t1.idtvarkarastis = ida and idstotele = dotid and
              t1.idatkarpa = t2.idatkarpa))/(t3.nuotolis+.000000000001),60),0)
  from admin.tlaikai as t1,admin.tatkarposestoteles as t2,
       admin.tatkarpos as t3
  where t1.idatkarpa = t2.idatkarpa and t1.idatkarpa = t3.id and
        t2.idstotele = dotid and t1.tipas in(0,2,4,8,9,12,16) and
        t1.idatkarpa = atkarpa and idtvarkarastis = ida and
        t1.idkryptis in(virziens,pretvirziens)
  order by isvyksta asc;
print 'Procedure getmarsh!';
open vw_cur;
vw_loop: loop
  fetch next vw_cur into ida, transp, marshnumber, marshname, virziens,
                         marshtypes, marshattribute;
  //nolasiits viens marshruts
```

```
  if(sqlstate = err_notfound) then
    leave vw_loop
  else
    //ja marshruti veel ir- nolasam marshruta preteejo virzienu
    if marshnumber not like '%*' then
      //galvenais marshruts
      select t2.id into pretvirziens
        from admin.tTvarkarasciai as t1,admin.tkryptys as t2
        where t2.tipas = 'B>A' and t1.id = t2.idtvarkarastis and t1.id = ida
    else
      //papildus marshruts
      select idpretkr from TAB_TEMPMARSH
        where idmarsh=ida and idkr=virziens;
    end if;
  end if;
print '%1!, %2!, %3!, %4!,%5!', ida, marshname, virziens, marshtypes,
  marshattribute;
insert into admin.tab_marshruts(number, name, type, attribute, tvarkarastis,
  transports)
  values(marshnumber, marshname, marshtypes, marshattribute,ida, transp);
//atrast tiko ierakstiita marshruta identifikatoru TAB_MARSHRUTS
select max(id_marshruts) into marshid
  from admin.tab_marshruts
  where type = marshtypes;
//atrast minimaalas mashinas id, kas kursee shii marshrutaa
select min(t1.idMasina) into minmas
  from admin.tlaikai as t1
  where t1.idtvarkarastis = ida and t1.idkryptis = virziens;
set dotplace=0;
set i=0;
open qq_cur;
fetch next qq_cur into xx,yy,dotname,dottype,dotnum,dotvirziens,dotid,
                       atkarpa;
//mekleet pirmo pieturu marshrutaa
gg_loop: loop
  if(dottype = 4) and(dotvirziens = virziens) then
    leave gg_loop
    //atrasts!!!
  end if;
  fetch next qq_cur into xx,yy,dotname,dottype,dotnum,dotvirziens,dotid,
                         atkarpa;
end loop gg_loop;
insert into admin.tab_marshcoord(marshruts,x,y,name,type,place,xrs,yrs,
  direction1) values(marshid,ROUND(xx*1000*2.5-727500,0),
  ROUND(yy*1000*(-2.5)+16387500,0),dotname,dottype,1,xx,yy,0);
select max(id_marshcoord) into marshcoordid
  from admin.tab_marshcoord where
  marshruts = marshid and xrs = xx and yrs = yy and type = dottype;
//time2 begin- mekleet atieshanas laikus no gp
open time2_cur;
time2_loop: loop
  fetch next time2_cur into timeh,timem;
  if(sqlstate = err_notfound) then
    leave time2_loop
  end if;
  //pieturas laiku atrashana
  if timem = 60 then
    set timeh=timeh+1;
    set timem=0
  end if;
  set timeh=mod(timeh,24);
  insert into admin.tab_marshtimestamps(times,marshcoord) values(
    timeh || ':' || timem,marshcoordid)
end loop time2_loop;
```

```
close time2_cur;
//time2 end
set namesak=dotname;
set prevdotnum='.';
set prevname='.';
fetch next qq_cur into xx,yy,dotname,dottype,dotnum,dotvirziens,dotid,
                       atkarpa;
qq_loop: loop
  set dotplace=dotplace+1;
  if dottype = 4 then
    select count(*) into checkp
      from admin.tKryptyseAtkarpos
      where idatkarpa = dotid and idkryptis = dotvirziens and pozymis = '0';
    if checkp = 0 then
      //atrasta pietura ar pozymis<>0, parbaudiit vai neatkartojas
      if((dotnum <> prevdotnum) and(dotname <> prevname)) then
        insert into admin.tab_marshcoord(marshruts,x,y,name,type,place,xrs,
          yrs) values(marshid,ROUND(xx*1000*2.5-727500,0),
          ROUND(yy*1000*(-2.5)+16387500,0),dotname,dottype,dotplace,xx,yy)
      else
        if(dotvirziens = pretvirziens) and(i = 0) then
          //atrasts otrs galapunkts
          insert into admin.tab_marshcoord(marshruts,x,y,name,type,place,
            xrs,yrs,direction2) values(marshid,ROUND(xx*1000*2.5-727500,0),
            ROUND(yy*1000*(-2.5)+16387500,0),dotname,dottype,dotplace,xx,
            yy,1);
          set i=1;
          select max(id_marshcoord) into marshcoordid
            from admin.tab_marshcoord
            where marshruts = marshid and xrs = xx and yrs = yy and
                  type = dottype;
          open time2_cur;
          time2_loop: loop
            fetch next time2_cur into timeh,timem;
            if(sqlstate = err_notfound) then
              leave time2_loop
            end if;
            if timem = 60 then
              set timeh=timeh+1;
              set timem=0
            end if;
            set timeh=mod(timeh,24);
            insert into admin.tab_marshtimestamps(times,marshcoord)
              values(timeh || ':' || timem,marshcoordid)
          end loop time2_loop;
          close time2_cur
        end if
      end if
    end if
  else
    //parasts punkts
    insert into admin.tab_marshcoord(marshruts,x,y,name,type,place,xrs,yrs)
       values(marshid,ROUND(xx*1000*2.5-727500,0),
       ROUND(yy*1000*(-2.5)+16387500,0),dotname,dottype,dotplace,xx,yy)
  end if;
  select max(id_marshcoord) into marshcoordid
    from admin.tab_marshcoord
    where marshruts = marshid and xrs = xx and yrs = yy and type = dottype;
  if dottype = 4 then
    if checkp = 0 then
      if((dotnum <> prevdotnum) and(dotname <> prevname)) then
        //atrasta vienkaarsha pietura, jaatrod laiks
        open time_cur;
        time_loop: loop
```

```
              fetch next time_cur into timeh,timem;
              if(sqlstate = err_notfound) then
                leave time_loop
              end if;
              if timem = 60 then
                set timeh=timeh+1;
                set timem=0
              end if;
              set timeh=mod(timeh,24);
              insert into admin.tab_marshtimestamps(times,marshcoord) values(
                timeh || ':' || timem,marshcoordid)
            end loop time_loop;
            close time_cur;
            set prevname=dotname;
            set prevdotnum=dotnum;
          end if
        end if
      end if;
      if(i = 1 and prevname = namesak) then
        //ja otrs galapunkts jau bija un pietura sakriit ar pirmo pieturu
        //marshruts ir pabeigts
        leave qq_loop
      end if;
      fetch next qq_cur into xx,yy,dotname,dottype,dotnum,dotvirziens,dotid,
                             atkarpa;
      if(sqlstate = err_notfound) then
        leave qq_loop
      end if
    end loop qq_loop;
    close qq_cur;
    call internet.getcoordlevels(marshid)
  end loop vw_loop;
  close vw_cur;
  update admin.tab_marshcoord set lvl = 3 where type = 4;
  call internet.getvienpiet();
  call internet.gettimes();
  call internet.addmarshsequence();
  call Internet.gettimesseq();
  call internet.addmarshtypes();
  call Internet.getdirections();
  //ierakstiit marshrutu preteejo virzienu nosukumus TAB_MARSHRUTS
  update admin.tab_marshruts set pretname = pavadinimas from
    tKryptys where tvarkarastis = idTvarkarastis and tipas = 'B>A';
  update admin.tab_marshruts pmarsh set pmarsh.mainmarsh=
    (select min gmarsh.id_marshruts
       from admin.tab_marshruts gmarsh
       where gmarsh.number=pmarsh.number||' *' and
             gmarsh.transports=pmarsh.transports)
  where pmarsh.number like '%*%' and mainmarsh is null;
  commit work;
  print 'Procedure getmarsh complete!';
  return
end
```

9.GETCOORDLEVELS procedure

```
ALTER procedure internet.getcoordlevels(in marsh varchar(10))
//Procedura aizpilda TAB_MARSHCOORD tabulas lvl lauku
begin
  declare err_notfound exception for sqlstate value '02000';
  declare xx1 real;
  declare yy1 real;
  declare xx2 real;
  declare yy2 real;
  declare ii1 numeric(9);
  declare ii2 numeric(9);
  declare lim integer;
  declare attalums real;
  declare minid numeric(9);
  declare maxid numeric(9);
  //marshruta punktu atlase
  declare vw_cur dynamic scroll cursor for
    select x,y,id_marshcoord from admin.tab_marshcoord
    where lvl = lim and marshruts = marsh;
  print 'Procedure getcoordlvls!';
  //atlauts attaalums starp punktiem pikselos
  set attalums=60;
  set lim=5;
  update admin.tab_marshcoord set lvl = 5 where marshruts = marsh;
  //atrast galapunktus
  select min(id_marshcoord),max(id_marshcoord) into minid,maxid
    from admin.tab_marshcoord
    where marshruts = marsh and type = 4;
  vv_loop: loop
    if lim > 0 then
      open vw_cur;
      fetch next vw_cur into xx1,yy1,ii1;
      if(sqlstate = err_notfound) then
        leave vv_loop
      end if;
      fetch next vw_cur into xx2,yy2,ii2;
      if(sqlstate = err_notfound) then
        leave vv_loop
      end if;
      update admin.tab_marshcoord set lvl = lim-1 where id_marshcoord = minid;
      vw_loop: loop
        // attalumu apreekinaashana
        if SQRT((xx2-xx1)*(xx2-xx1)+(yy2-yy1)*(yy2-yy1)) > attalums then
          update admin.tab_marshcoord set lvl = lim-1 where id_marshcoord = ii2;
          set xx1=xx2;
          set yy1=yy2;
          set ii1=ii2;
          fetch next vw_cur into xx2,yy2,ii2;
          if(sqlstate = err_notfound) then
            leave vw_loop
          end if
        else
          fetch next vw_cur into xx2,yy2,ii2;
          if(sqlstate = err_notfound) then
            leave vw_loop
          end if
        end if
      end loop vw_loop;
      set attalums=attalums*5;
      set lim=lim-1
    else
      leave vv_loop;
```

```
      close vw_cur
    end if;
    close vw_cur
  end loop vv_loop;
  update admin.tab_marshcoord set lvl = 0 where id_marshcoord = maxid;
  return
end
```

10.GETVIENPIET procedure

```
ALTER procedure internet.Anny_getvienpiet()
//Procedura aizpilda TAB_MARSHCOORD tabulas equal lauku
begin
  declare err_notfound exception for sqlstate value '02000';
  declare i integer;
  declare ida numeric(9);
  //kursors, kas atlasa pieturas
  declare vw_cur dynamic scroll cursor for
    select id_marshcoord,newname,x,y
      from admin.tab_marshcoord
      where type is not null and newname is not null
      order by newname asc,x asc,y asc;
  declare pietname varchar(50);
  declare pietx real;
  declare piety real;
  declare prevpietx real;
  declare prevpiety real;
  declare prevpietname varchar(50);
  print 'Procedure getvienpiet!';
  set i=1;
  update admin.tab_marshcoord set equal = null;
  open vw_cur;
  fetch next vw_cur into ida,prevpietname,prevpietx,prevpiety;
  //pirmaa pietura
  update admin.tab_marshcoord set equal = 1 where id_marshcoord = ida;
  vw_loop: loop
    fetch next vw_cur into ida,pietname,pietx,piety;
    if(sqlstate = err_notfound) then
      leave vw_loop
    end if;
    //vai pietura sakriit ar iepriekshēejo
    if prevpietname = pietname and abs(prevpietx-pietx) < 100 and
       abs(prevpiety-piety) < 100 then
      update admin.tab_marshcoord set equal = i where id_marshcoord = ida
    else
      //nosaukums nesakriit vai attaalums ir lielaaks par atlauto
      set i=i+1;
      update admin.tab_marshcoord set equal = i where id_marshcoord = ida;
      set prevpietname=pietname;
      set prevpietx=pietx;
      set prevpiety=piety
    end if
  end loop vw_loop;
  close vw_cur
end
```

11.GETTIMES procedure

```
ALTER procedure Internet.Anny_gettimes()
//Procedura aizpilda TAB_MARSHTIME tabulu saskanaa ar TAB_MARSHTIMESTAMPS datiem
begin
  declare err_notfound exception for sqlstate value '02000';
  //kursors kas atlasa pieturas
  declare time_cur dynamic scroll cursor for
    select id_marshcoord,marshruts,attribute,x,y,equal
      from admin.tab_marshcoord,admin.tab_marshruts
      where admin.tab_marshcoord.type is not null and marshruts = id_marshruts;
  //kursors kas atlasa laikus
  declare hhh_cur dynamic scroll cursor for
    select mainmarsh,minute(times)
      from admin.tab_marshtimestamps,admin.tab_marshcoord,admin.tab_marshruts
        where hour(times) = hhour and marshcoord = id_marshcoord and
         marshruts = id_marshruts and equal = marsheq and
         (marshcoord = marshcoordid or(abs(x-marshcoordx) <= 25 and
         abs(y-marshcoordy) <= 25 and mainmarsh = idmain and attribute = attr))
      order by times asc;
  declare attr varchar(50);
  declare hhour integer;
  declare mminute varchar(4);
  declare minutest varchar(600);
  declare marshcoordid numeric(9);
  declare marshtimeid numeric(9);
  declare idmain numeric(9);
  declare i integer;
  declare cnt numeric(9);
  declare mmm numeric(9);
  declare marshcoordx real;
  declare marshcoordy real;
  declare marsheq numeric(9);
  set i=0;
  set hhour=0;
  //ierakstu skaits
  select count(*) into cnt from admin.tab_marshcoord,admin.tab_marshruts where
    admin.tab_marshcoord.type is not null and marshruts = id_marshruts;
  open time_cur;
  time_loop: loop
    fetch next time_cur into marshcoordid,idmain,attr,marshcoordx,marshcoordy,
      marsheq;
    if(sqlstate = err_notfound) then
      leave time_loop
    end if;
    set i=i+1;
    //izvadam apstraadaato pieturu skaitu un kopiigo skaitu
    if mod(i,50) = 0 then
      print 'gettimes: Apstr. i=%1!/%2! ieraksti!',i,cnt
    end if;
    //jaunaas rindas iestarpinaashana TAB_MARSHTIME
    insert into admin.tab_marshtime(id_marshtime,marshcoord)
      values(i,marshcoordid);
    if(select count(*) from admin.tab_marshtimestamps where
        marshcoord = marshcoordid) > 0 then
      set hhour=0;
      set minutest='';
      while hhour <> 24 loop
        set minutest='';
        open hhh_cur;
        hhh_loop: loop
          //laiku apstraade
          fetch next hhh_cur into mmm,mminute;
```

```
  if(sqlstate = err_notfound) then
    leave hhh_loop
  end if;
  if length(mminute) = 1 then set mminute='0' || mminute
  end if;
  set mmm=ifnull(mmm,0,mmm);
  if mmm > 0 then
    set mminute='*' || mminute || '!'
  end if;
  set minutest=minutest || ' ' || mminute
end loop hhh_loop;
close hhh_cur;
if hhour = 0 then
  update admin.tab_marshtime set h0 = minutest
    where id_marshtime = i;
  set hhour=hhour+1
else
  //atshkiriibaa no hhour veertiibas aizpildam h_ lauku
  if hhour = 1 then
    update admin.tab_marshtime set h1 = minutest
      where id_marshtime = i;
    set hhour=hhour+1
  else
    if hhour = 2 then
      update admin.tab_marshtime set h2 = minutest
        where id_marshtime = i;
      set hhour=5
    else
      if hhour = 5 then
        update admin.tab_marshtime set h5 = minutest
          where id_marshtime = i;
        set hhour=hhour+1
      else
        if hhour = 6 then
          update admin.tab_marshtime set h6 = minutest
            where id_marshtime = i;
          set hhour=hhour+1
        else
          if hhour = 7 then
            update admin.tab_marshtime set h7 = minutest
              where id_marshtime = i;
            set hhour=hhour+1
          else
            if hhour = 8 then
              update admin.tab_marshtime set h8 = minutest
                where id_marshtime = i;
              set hhour=hhour+1
            else
              if hhour = 9 then
                update admin.tab_marshtime set h9 = minutest
                  where id_marshtime = i;
                set hhour=hhour+1
              else
                if hhour = 10 then
                  update admin.tab_marshtime set h10 = minutest
                    where id_marshtime = i;
                  set hhour=hhour+1
                else
                  if hhour = 11 then
                    update admin.tab_marshtime set h11 = minutest
                      where id_marshtime = i;
                    set hhour=hhour+1
                  else
                    if hhour = 12 then
```

```
      update admin.tab_marshtime set h12 = minutest
        where id_marshtime = i;
      set hhour=hhour+1
    else
      if hhour = 13 then
        update admin.tab_marshtime set h13 = minutest
          where id_marshtime = i;
        set hhour=hhour+1
      else
        if hhour = 14 then
          update admin.tab_marshtime set h14 = minutest
            where id_marshtime = i;
          set hhour=hhour+1
        else
          if hhour = 15 then
            update admin.tab_marshtime
              set h15 = minutest
              where id_marshtime = i;
            set hhour=hhour+1
          else
            if hhour = 16 then
              update admin.tab_marshtime
                set h16 = minutest
                where id_marshtime = i;
              set hhour=hhour+1
            else
              if hhour = 17 then
                update admin.tab_marshtime
                  set h17 = minutest
                  where id_marshtime = i;
                set hhour=hhour+1
              else
                if hhour = 18 then
                  update admin.tab_marshtime
                    set h18 = minutest
                    where id_marshtime = i;
                  set hhour=hhour+1
                else
                  if hhour = 19 then
                    update admin.tab_marshtime
                      set h19 = minutest
                      where id_marshtime = i;
                    set hhour=hhour+1
                  else
                    if hhour = 20 then
                      update admin.tab_marshtime
                        set h20 = minutest
                        where id_marshtime = i;
                      set hhour=hhour+1
                    else
                      if hhour = 21 then
                        update admin.tab_marshtime
                          set h21 = minutest
                          where id_marshtime = i;
                        set hhour=hhour+1
                      else
                        if hhour = 22 then
                          update admin.tab_marshtime
                            set h22 = minutest
                            where id_marshtime = i;
                          set hhour=hhour+1
                        else
                          if hhour = 23 then
                            update admin.tab_marshtime
```

```
                                                    set h23 = minutest
                                                    where id_marshtime = i;
                                                  set hhour=hhour+1
                                                end if
                                              end if
                                            end if
                                          end if
                                        end if
                                      end if
                                    end if
                                  end if
                                end if
                              end if
                            end if
                          end if
                        end if
                      end if
                    end if
                  end if
                end if
              end if
            end if
          end if
        end if
      end loop
    end if
  end loop time_loop;
  close time_cur
end
```

12.ADDMARSHSEQUENCE procedure

```
ALTER procedure internet.addmarshsequence()
//Procedura aizpilda TAB_MARSHRUTS tabulas sequence lauku
begin
  declare err_notfound exception for sqlstate value '02000';
  declare ida numeric(9);
  declare nmn varchar(32);
  declare seq varchar(32);
  declare k integer;
  declare seqint integer;
  //kursors kas atlasa marshrutus
  declare qq_adr dynamic scroll cursor for
    select id_marshruts,number
    from admin.tab_marshruts
    order by id_marshruts asc;
  declare sub varchar(64);
  open qq_adr;
  qq_loop: loop
    fetch next qq_adr into ida,nmn;
    if(sqlstate = err_notfound) then
      leave qq_loop
    end if;
    //marshruta numura paarveidoshana
    set seq=internet.normalizetxt(nmn);
    set seq=replace(seq,'a','');
    set seq=replace(seq,'b','');
    set seq=replace(seq,'c','');
    set seq=replace(seq,'d','');
    set seq=replace(seq,'e','');
    set seq=replace(seq,'f','');
    set seq=replace(seq,'g','');
    set seq=replace(seq,'h','');
    set seq=replace(seq,'i','');
    set seq=replace(seq,'j','');
    set seq=replace(seq,'k','');
    set seq=replace(seq,'l','');
    set seq=replace(seq,'m','');
    set seq=replace(seq,'n','');
    set seq=replace(seq,'o','');
    set seq=replace(seq,'p','');
    set seq=replace(seq,'q','');
    set seq=replace(seq,'r','');
    set seq=replace(seq,'s','');
    set seq=replace(seq,'t','');
    set seq=replace(seq,'u','');
    set seq=replace(seq,'v','');
    set seq=replace(seq,'w','');
    set seq=replace(seq,'x','');
    set seq=replace(seq,'y','');
    set seq=replace(seq,'z','');
    set seq=replace(seq,'-','.');
    set seq=replace(seq,'/','.');
    set seq=replace(seq,'(','.');
    if seq like '%.%' then
      set k=1;
      set sub='';
      //nolasam liidz pirmajam punktam
      while substring(seq,k,1) <> '.' and k < length(seq) loop
        set sub=sub || substring(seq,k,1);
        set k=k+1
      end loop;
      set seq=sub
```

```
    end if;
    set seqint=convert(integer,seq);
    update admin.tab_marshruts set sequence = seqint where id_marshruts = ida
  end loop qq_loop;
  close qq_adr;
  print 'Done internet.addmarshsequence!';
  return
end
```

13.ADDMARSHTYPES procedure

```
ALTER procedure internet.addmarshtypes()
//Procedura aizpilda TAB_MARSHRUTS tabulas marshtype lauku
begin
  declare err_notfound exception for sqlstate value '02000';
  declare ida numeric(9);
  declare nmn varchar(32);
  declare k integer;
  declare i integer;
  declare tips integer;
  declare tr integer;
  //kursors kas nolasa marshrutus
  declare qq_adr dynamic scroll cursor for
    select id_marshruts,number,transports
     from admin.tab_marshruts where marshtype is null
     order by id_marshruts asc;
  update admin.tab_marshruts set marshtype = null;
  open qq_adr;
  qq_loop: loop
    fetch next qq_adr into ida,nmn,tr;
    if(sqlstate = err_notfound) then
      leave qq_loop
    end if;
    //cik reizes marshruts atkaartojas TAB_MARSHRUTS tabulaa
    select count(*) into i from admin.tab_marshruts
      where number = nmn and transports = tr;
    if i = 3 then
      set tips=3
    else
      if i = 2 then
        if(select count(*) from admin.tab_marshruts
           where number = nmn and transports = tr and attribute = '6') <> 0 then
          set tips=4
        else
          if(select count(*) from admin.tab_marshruts
             where number = nmn and transports = tr and attribute = '6-7') <> 0
            then
            set tips=2
          end if
        end if
      else
        if(select count(*) from admin.tab_marshruts
           where number = nmn and transports = tr and attribute = '1-7') <> 0
          then
          set tips=1
        else
          if(select count(*) from admin.tab_marshruts
             where number = nmn and transports = tr and attribute = '1-5') <> 0
            then
            set tips=5
          else
            set tips=6
          end if
        end if
      end if
    end if;
    update admin.tab_marshruts set marshtype = tips where id_marshruts = ida
  end loop qq_loop;
  close qq_adr;
  print 'Done internet.addmarshtypes!';
  return
end
```

14.GETDIRECTIONS procedure

```
ALTER procedure internet.getdirections()
//Procedura aizpilda TAB_MARSHCOORD tabulas direction1 un directons2 laukus
begin
  declare err_notfound exception for sqlstate value '02000';
  declare minid numeric(9);
  declare marsh integer;
  //kursors kas atlasa marshrutu identifikatorus
  declare vw_cur dynamic scroll cursor for
    select id_marshruts from admin.tab_marshruts;
  open vw_cur;
  vv_loop: loop
    fetch next vw_cur into marsh;
    if(sqlstate = err_notfound) then
      leave vv_loop
    end if;
    //atrast videejo punktu, galapunktu
    select min(id_marshcoord) into minid
      from admin.tab_marshcoord where
      marshruts = marsh and direction2 = 1;
    update admin.tab_marshcoord set direction1 = 1
      where id_marshcoord < minid and marshruts = marsh and direction1 is null;
    update admin.tab_marshcoord set direction1 = 2
      where id_marshcoord >= minid and marshruts = marsh and direction1 is null;
    //pirmais punkts, galapunkts
    update admin.tab_marshcoord set direction2=1
      where id_marshcoord=
      (select min(id_marshcoord) where type=4 and marshruts=marsh);
    update admin.tab_marshcoord set direction2=0
      where marshruts=marsh and direction2 is null
  end loop vv_loop;
  return
end
```

15.FINDROUTE function

```
ALTER function internet.findroute(in int1 numeric(9),in int2 numeric(9),in tips
integer)
//Funkcija, kas atrod marshrutu no punkta A liidz punktam B
returns varchar(256)
begin
  declare marshr numeric(9);
  declare retstring varchar(256);
  declare point1 numeric(9);
  declare point2 numeric(9);
  print 'Procedure internet.Anny_findroute!';
  select equal into point1
    from tab_marshcoord
    where id_marshcoord=int1;
  select equal into point2
    from tab_marshcoord
    where id_marshcoord=int2;
  //mekleet kopiigo marshrutu
  if tips = 1 then
    select top 1 m1.marshruts into marshr
      from admin.tab_marshcoord as m1,admin.tab_marshcoord as m2,
           admin.tab_marshruts as s1
      where m1.equal = point1 and m2.equal = point2 and
            m1.marshruts = m2.marshruts and m1.marshruts = s1.id_marshruts and
            s1.attribute in('1-7','1-5')
  else
      if tips = 6 then
        select top 1 m1.marshruts into marshr
          from admin.tab_marshcoord as m1,admin.tab_marshcoord as m2,
               admin.tab_marshruts as s1
          where m1.equal = point1 and m2.equal = point2 and
                m1.marshruts=m2.marshruts and m1.marshruts = s1.id_marshruts and
                s1.attribute in('6','6-7','1-7')
    else
      select top 1 m1.marshruts into marshr
        from admin.tab_marshcoord as m1,admin.tab_marshcoord as m2,
             admin.tab_marshruts as s1
        where m1.equal = point1 and m2.equal = point2 and
              m1.marshruts = m2.marshruts and m1.marshruts = s1.id_marshruts and
              s1.attribute in('7','6-7','1-7')
    end if
  end if;
  if marshr <> '' then
    //eksistee kopiigs marshruts
    set retstring=marshr
  else
    //ja marshruts netika atrasts
    set retstring='There is no route!';
    if tips = 1 then
      //pieturas A marshrutu saraksts
      select distinct marshruts into #pointmarsh1
        from admin.tab_marshcoord,admin.tab_marshruts
        where equal = point1 and marshruts = id_marshruts and
              attribute in('1-5','1-7');
      //pieturas B marshrutu saraksts
      select distinct marshruts into #pointmarsh2
        from admin.tab_marshcoord,admin.tab_marshruts
        where equal = point2 and marshruts = id_marshruts and
              attribute in('1-5','1-7')
    else
      if tips = 6 then
        select distinct marshruts into #pointmarsh1
```

```
          from admin.tab_marshcoord,admin.tab_marshruts
            where equal = point1 and marshruts = id_marshruts and
                  attribute in('6','6-7','1-7');
        select distinct marshruts into #pointmarsh2
          from admin.tab_marshcoord,admin.tab_marshruts
          where equal = point2 and marshruts = id_marshruts and
                attribute in('6','6-7','1-7')
      else
        select distinct marshruts into #pointmarsh1
          from admin.tab_marshcoord,admin.tab_marshruts
          where equal = point1 and marshruts = id_marshruts and
                attribute in('7','6-7','1-7');
        select distinct marshruts into #pointmarsh2
          from admin.tab_marshcoord,admin.tab_marshruts
          where equal = point2 and marshruts = id_marshruts and
          attribute in('7','6-7','1-7')
      end if
    end if;
    select top 1 p1.marshruts || ' ' || p2.marshruts || ' pietura ' || sequence
      into retstring
      from admin.tab_marshruts_sequence,#pointmarsh1 as p1,#pointmarsh2 as p2
      where p1.marshruts = marshruts1 and p2.marshruts = marshruts2 and
            sequence is not null
  end if;
  print 'Done internet.Anny_findroute!';
  return retstring
end
```

Bakalaura darbs izstrādāts
LU Datorikas nodaļā

Autors:
Fizikas un matemātikas . . . . . .

fakultātes studente Anna Jevtušenko
St. apl. Nr. DatZ020013

Darba vadītājs
M. Dat Ija Šaporenkova. . . . . .

Recenzents

Darbs iesniegts Datorikas nodaļā 2006. g. . . . . maijā.

Pieņēma sekretāre . . . . . .

Aizstāvēts datorzinātņu bakalaura pārbaudījumu komisijas sēdē

2006.g. . . . . .ar atzīmi. . . . . .

Protokols Nr._____

Bakalaura pārbaudījumu .. . .
Komisijas sekretārs