KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Giedrius Racibara

DINAMINĖ GEOGRAFINIŲ IR ATRIBUTINIŲ DUOMENŲ SĄSAJA ORACLE DBVS PAGRINDU

Magistro darbas

Darbo vadovas
prof. dr. R. Butleris

KAUNAS, 2009

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

DINAMINĖ GEOGRAFINIŲ IR ATRIBUTINIŲ DUOMENŲ SĄSAJA ORACLE DBVS PAGRINDU

Magistro darbas

Vadovas
prof. dr. R. Butleris
2009-01-

Konsultantas
lekt. T. Danikauskas

Recenzentas
dr. S. Drąsutis
2009-01-

Atliko
IFM 3/4 gr. stud.
Giedrius Racibara
2009-01-05

KAUNAS, 2009

Turinys

Įvadas

Sėkmingam verslo funkcionavimui nepakanka vien sukaupto didelio duomenų kiekio apie specifinę sritį. Efektyvus operavimas duomenimis ir greitas bei aiškus rezultatų pateikimas yra labai svarbi daugelio įvairių sričių įmonių problema. Nenaudojant specialių priemonių, tai pasiekti ganėtinai sunku. Erdvinės informacijos analizės naudą pripažįsta ir vis labiau vertina verslo įmonės ir organizacijos, per GIS(Geografines informacines sistemas) technologijas atrasdamos naujus verslo sprendimus, klientus ir tolimesnės plėtros galimybes. Geografinės informacinės sistemos (GIS) šiuo metu yra plačiai taikomos verslo problemų sprendimuose, tačiau jų integravimas į egzistuojančias ar naujas sistemas yra sudėtingas, bei daug lėšų reikalaujantis procesas.[1] [2] [3]

Dažnai įmonėms nepakanka vien galimybės naudotis elektroniniais žemėlapiais, daugeliu atveju savo surinktus duomenis apie specifinę sritį, norima matyti žemėlapyje keletu pjūvių. Paprastai įmonė turi tik duomenis apibūdinančius jų veiklos sritį, o GIS paslaugas teikiančios įmonės turi didelius kiekius erdvinių duomenų, bet nieko nežino apie duomenis, kuriuos turi verslo įmonės. Stengiantis perkelti ir analizuoti verslo duomenis ant skaitmeninio žemėlapio, atsiranda problemos, kadangi GIS paslaugų tiekėjai duoda tik žemėlapio valdymo galimybes, o įmonės samdo programuotojus, kad šie sukurtų sudėtingas taikomąsias programas skirtas verslo duomenų analizei ir jų rodymui žemėlapyje. Paprastai skirtingoms verslo sritims reikalinga skirtinga jų sukauptų duomenų analizė ir sukaupti duomenys skiriasi iš esmės. Todėl kiekvienam tokiam atvejui samdomi nauji programuotojai ir kuriamos naujos taikomosios programos atitinkančios skirtingų verslo šakų skirtingus poreikius. [3]

Žvelgiant į esamą situaciją yra akivaizdu, kad įmonės sutaupytų daug pinigų, jei kiekvieną kartą norint joms matyti ir analizuoti savo surinktus duomenis žemėlapyje, nerikėtų samdyti programuotojų, kad šie sukurtų naujas taikomąsias programas skirtas specifinei sričiai.

Per paskutinius dešimtį metų turėjome daugybę GIS taikomųjų programų pavyzdžių, kurie užtikrino sėkmingą erdvinės informacijos produktų pateikimą tiek viešuosiuose tiek ir privačiuose sektoriuose. Neabejotinai šios sistemos buvo naudingos, tačiau dauguma organizacijų vis dar susiduria su GIS taikomųjų programų diegimo savo veiklos srityje problemomis.[1] [3] [5]

Kuriant geografines informacines sistemas yra naudinga skaidyti jų architektūrą atskiromis dalimis, kurios gali būti patalpinamos atskiruose nutolusiuose serveriuose. Tokiu būdu galima atskirti erdvinių duomenų saugojimo, apdorojimo ir pritaikymo individualioms klientų reikmėms logiką. Dauguma geografinių informacinių sistemų yra sukurtos naudojant trijų dalių architektūrą kurią sudaro[3]:

- Duomenų laikymas
- Taikomosios paslaugos
- Kliento taikomosios programos

Ši architektūra leidžia atlikti ankščiau minėtą geografinės informacinės sistemos dalių išskaidymą atskiruose serveriuose. Architektūros išskaidymas leidžia sistemai veikti greičiau ir sumažina gedimų tikimybę, kadangi identifikavus dalį kurioje yra problemų galima ją greitai atnaujinti ar pakeisti kita, neliečiant likusių sistemos dalių. Toks išskaidymas taip pat leidžia įmonėms sumažinti investicijas į erdvinius žemėlapius, kadangi joms nereikia turėti erdvinių duomenų savo žinioje. Erdvinė informacija iš kurios yra generuojami žemėlapiai yra saugoma vienoje vietoje ir už juos atsako įmonė teikianti žemėlapių paslaugas, todėl įmonėms klientėms tereikia išsipirkti licenciją žemėlapių panaudojimui, neperperkant visų erdvinių duomenų.

Analizuojant egzistuojančias GIS sistema reikia atsižvelgti kaip jos išpildo minėtą trijų dalių architektūrą. Tokiu būdu lyginant jas viena su kita galima identifikuoti pagrindinius trūkumus ir iškelti kriterijus kurių esančios GIS sistemos netenkina bei surasti priežastis kodėl įmonėms reikia naujų GIS sprendimų, kurie būtų lankstesni ir atitiktų įmonių lūkesčius už minimalią kainą. Pagrindiniai kriterijai pagal kuriuos analizuojant reikėtų vertinti esamus GIS sprendimus yra šie:

- Įmonės aprašomųjų duomenų atvaizdavimas žemėlapyje – ar yra galimybė naudojantis esamais GIS sprendimais savo specifinius duomenis matyti žemėlapyje (pvz.: matyti parduotuves žemėlapyje, kur įmonės apyvarta yra 50% didesnė nei pernai)
- Programavimo darbai – kiek investicijų reikia skirti programavimo darbams, kad įmonės aprašomieji duomenys būtų vaizduojami žemėlapyje
- Panaudojimo paprastumas – įmonės aprašomųjų duomenų vaizdavimui žemėlapyje turi reikėti minimalių pastangų.
- Minimalios investicijos – investicijos į GIS sistemos panaudojimą savo versle turi būti minimalios

Kuriant geografinę informacinę sistemą viena svarbiausių dalių yra erdvinių duomenų saugojimo ir apdorojimo galimybė. Kadangi erdviniai duomenys, remiantis trijų dalių architektūra[3], turi būti saugomi vienoje vietoje, reikia užtikrinti greitą ir efektyvų šios informacijos gavimą ir įrašymą į duomenų bazę. Tuo tikslu reikia parinkti tinkamą duomenų bazių valdymo sistemą, kuri užtikrintų šį mechanizmą. Siekiant sukurti naują GIS sprendimą,

darbe bus analizuojama Oracle programinė įranga. Šios įrangos paskirtis - saugojimas bei operavimas erdviniais duomenimis, kurie talpinami vienoje nutolusioje duomenų bazėje.

Oracle yra duomenų bazių valdymo sistemų rinkos lyderis[5] [6]. Oracle 10g duomenų bazių valdymo sistema turi erdvinės informacijos saugojimo bei apdorojimo galimybes. Analizuojant Oracle programinę įrangą reikia identifikuoti funkcijas kuriomis galima pasinaudoti operuojant erdviniais duomenimis bei identifikuoti Oracle programinės įrangos komponentus, kurie leistų iš erdvinės informacijos generuoti skaitmeninius žemėlapius.

Apibendrinant - šio darbo tikslas yra išanalizuoti esamas GIS sistemas ir atsižvelgus į jų trūkumus bei verslo poreikius pateikti naujos GIS sistemos sprendimą, kurios architektūra realizuota naudojant Oracle programinę įrangą sukūrus trūkstamus komponentus.

Darbo struktūra:

1. Analizės etape yra analizuojami esami GIS sprendimai pateikiant jų trūkumus ir privalumus. Taip pat atliekama Oracle programinės įrangos analizė ieškant komponentų kurie leistų operuoti erdviniais duomenimis ir generuoti skaitmeninius žemėlapius.
2. Analizės rezultatų etape yra pateikiamas analizuotų esamų GIS sistemų palyginimas ir nurodomi jų trūkumai. Atsižvelgus į trūkumus yra pasiūlomas naujos, dinamiškos GIS sistemos sprendimas.
3. Projektavimo etape yra suprojektuojama siūlomo sprendimo architektūra, kuri sudaroma remiantis Oracle programinės įrangos komponentais. Naujoje siūlomos GIS sistemos architektūroje yra trūkstamas komponentas kurio veikimas yra pagrįstas veiklos taisyklėmis, todėl projektavimo etape yra pateiktas šio komponento projektas kuris yra realizuotas darbo metu.
4. Sistemos komponentų derinimo dalyje aprašoma Oracle programinės įrangos suderinamumas su realizuotu duomenų integravimo komponentu.
5. Testavimo dalyje aprašomas naujos dinamiškos GIS sistemos su duomenų integravimo komponentu testavimo procesas.
6. Darbo rezultatų apibendrinimo dalyje yra pateiktas sukurtos dinamiškos GIS sistemos palyginimas su analizuotais esamais sprendimais.
7. Darbo pabaigoje yra pateikiamos išvados

Darbo tema padarytas pranešimas „Dinamiškas erdvinių ir verslo duomenų integravimas panaudojant Oracle DBVS“, pristatytas 2008 metais vykusioje tarpuniversitetinėje magistrantų ir doktorantų konferencijoje „Informacinės Technologijos ‘2008“. Straipsnis išspausdintas konferencijos leidinyje.

Tarptautinei konferencijai „Information technologies 2009“ yra rengiamas pranešimas „Dynamic Integration of Geo spatial and Enterprise data in Information Systems“

1. ORACLE DBVS IR ESAMŲ GIS SPRENDIMŲ ANALIZĖ

1.1.ANALIZĖS TIKSLAS

Šios analizės tikslas – įvertinti esamą situaciją GIS taikomųjų programų srityje ir pasiūlyti savo sprendimą, esamiems GIS trūkumams spręsti. Siūlant savo sprendimą, reikia atsižvelgti į tai, kas yra atlikta šioje srityje, todėl tikslinga atlikti problemos sprendimo metodų literatūros šaltiniuose bei panašių sistemų, architektūros ir galimų ORACLE DBVS įgyvendinimo priemonių variantų analizę. Analizės rezultatai turėtų atskleisti dabartinies GIS integravimo į šiuolaikinio verslo procesus problemas, privalumus ir lūkesčius, bei pateikti sprendimo būdą kuris turėtų pagerinti GIS ir mažų bei vidutinių įmonių veiklos integravimo sprendimus.

Atlikus analizę, bus suformuluotas aiškus būsimos sistemos aprašymas, aiškiai suformuluoti būsimos sistemos tikslai, įvertinami būsimi darbo rezultatai ir jų pranašumas palyginti su išanalizuotais esamais sprendimais. Išanalizavus Oracle DBVS teikiamas galimybes bus įvertinti ir parinkti Oracle programinės įrangos komponentai, kurie yra reikalingi būsimam sprendimui įgyvendinti.

1.2. TYRIMO SRITIS, OBJEKTAS IR PROBLEMA

Tyrimo sritis - geografinės informacinės sistemos (GIS) kurios, šiuo metu yra sparčiai integruojamos daugelio įmonių verslo procesuose. Efektyvus šių sistemų panaudojimas tampa vienu svarbiausių šiuolaikinio verslo uždavinių. Daugelis įmonių yra sukaupę didelius kiekius jiems būdingų specifinių duomenų, kurie gali būti panaudoti daugelyje specifinių sričių (miestų planavimas, rinkodara, kartografija, kriminologija ir t.t.), todėl šią informaciją reikia greitai ir efektyviai susieti su erdvine informacija, bei pateikti vartotojui aiškioje ir patogioje formoje, t.y. skaitmeniniame žemėlapyje.

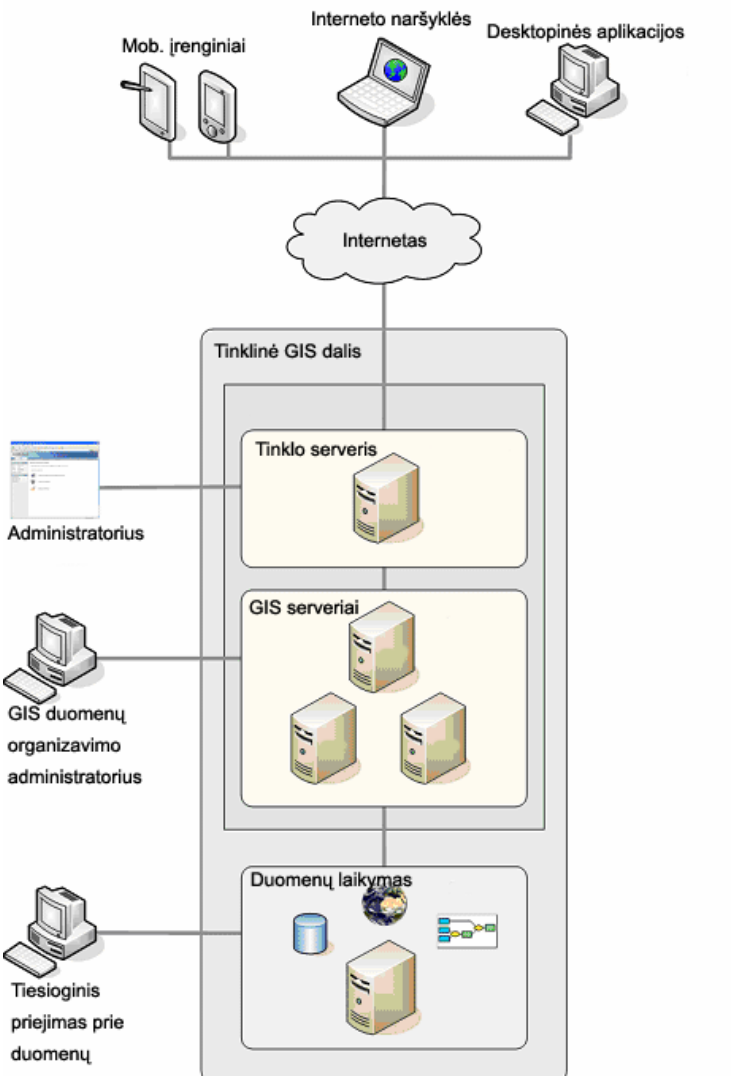
Dabartiniu metu daugelis įmonių turimų vietovės duomenų, yra nepanaudojami pagal galimybes ir galimą paskirtį. Tokie duomenys gali būti kliento adresai, bei kita dalykinė informacija turinti sąsają su erdviniais duomenimis. Įmonės dažnai turi spręsti klausimus, pagrįstus vietovės ryšiais tokiais kaip pardavimų kiekis kokioje nors teritorijoje, aptarnaujančio personalo vieta artima aptarnavimo taškui, ar netgi pirmo atsiliepiančiojo į SOS signalą radimas. Įtraukdamos vietovės atpažinimo galimybes į pagrindines elektroninio verslo funkcijas, organizacijos gali priimti geresnius sprendimus ir aptarnauti klientus žymiai efektyviau. [1] [3] [5]

Geografinė informacinė sistema (GIS) – informacinė sistema skirta darbui su erdvine ir aprašomąja informacija. Sistema skirta skaitmeninių koordinuotų erdvėje duomenų kaupimui, saugojimui, vaizdavimui, redagavimui, integravimui bei analizei[2].

GIS sudaro programinė įranga kuri jungia geografinę informaciją (kur daiktai yra) su aprašomąja informacija (į ką daiktai panašūs). Skirtingai nuo plokščio popierinio žemėlapio, GIS gali turėti daug sluoksnių informacijos po jo paviršiumi. Tokiu būdu galima matyti struktūrą ar tendencijas jūsų verslui būdingoje informacijoje, rodyti ir apskaičiuoti optimalius transporto priemonės maršrutus, sukurti norimas linijas, apskaičiuoti lygias lenktyniavimo sritis, identifikuoti naujas rinkas, nustatyti vietą konkurento svetainių, dirbti su medžiaga ir įrangos ataskaitomis nuo intuityvios grafinės vartotojo sąsajos, ir daug daugiau.

Tipinę bet kokios GIS sistemos architektūrą sudaro trys skirtingos dalys[3][4]:

- Duomenų Laikymas – apima įvairių erdvinių ir ne erdvinių duomenų sandėliavimą organizacijos viduje. Šitie duomenų sandėliai paprastai turi savyje informaciją keliuose nesuderintuose duomenų formatuose.
- Taikomosios Paslaugos - susideda iš GIS Taikomojo Serverio ir Tinklo Serverio, kurie yra atsakingi už duomenų prieigą, apdirbimą ir vartotojo aptarnavimą per kliento taikomąsias programas. Tinklo Serveris yra bet koks GIS Tinklo Serveris kuris leidžia komunikuoti su vartotojų standartiniais interneto serveriais tokiais kaip Apache, Microsoft IIS ir pan. Taikomasis GIS Serveris yra funkcinis pagrindas bet kokių GIS paslaugų jis suteikia bet kokių erdvinių duomenų integravimą ir transformavimą.
- Kliento taikomosios programos – klientais gali būti bet kuri taikomoji programa, geografiškai nutolusi nuo GIS paslaugas teikiančio serverio. Klientai paprastai kreipiasi į GIS taikomąsias paslaugas prašydami nupiešti tam tikrą žemėlapio dalį ir objektus ant jo. Tai gali būti interneto naršyklės, mobilieji įrenginiai ir t.t.



1 pav. GIS sistemos architektūra [3]

Ši trijų dalių architektūra yra ideali įmonės sistemoms, todėl, kad ji įgalina portatyvių ir išplečiamų sistemų konstravimą. Ji leidžia bet kuriai GIS sistemai supaprastinti skirtingų duomenų šaltinių integraciją, suteikiant funkcionalumo karkasą, kuris leidžia keistis žemėlapiais ir duomenimis tinkle.[3]

Sprendžiama problema – GIS panaudojimas versle yra sudėtingas ir daug lėšų kainuojantis procesas. Erdvinės ir specifinės verslo sukauptos informacijos integracija siekiant pateikti rezultatus žemėlapyje, šiuo metu nėra dinamiškas ir pakartotinai panaudojamas procesas.[1]

Tyrimo objektas yra Oracle DBVS. Oracle duomenų bazių valdymo sistemos (DBVS) teikiamų geografinių informacinių sistemų (GIS) kūrimo priemonės gali būti panaudotos

minėtai problemai spręsti. Reikia ištirti atskirų Oracle DBVS komponentų funkcionalumą ir tinkamumą GIS sistemų kūrimui.

1.3. TYRIMO TIKSLAS IR UŽDAVINIAI

Tyrimo tikslas yra ištirti esamų GIS sprendimų verslui trūkumus ir pateikti naują sprendimą, kuris bus pagrįstas Oracle DBVS programinės įrangos architektūra. Sėkmingam sprendimui realizuoti, šiame tyrime iškeliam tokius uždavinius:

- Suformuluoti būsimos sistemos sprendimo naujumą lyginant su egzistuojančiomis GIS sistemomis ir jų trūkumais.
- Išanalizuoti Oracle DBVS ir parinkti tinkamus komponentus būsimo sprendimo architektūrai realizuoti
- Identifikuoti ir suprogramuoti trūkstamus komponentus
- Parengti demonstracinius erdvinius duomenis
- Parengti demonstracinius skirtingas verslo sritis atspindinčius aprašomuosius duomenis.
- Pademonstruoti sprendimo naujumą ir efektyvumą.
- Suprojektuotos architektūros realizacijoje naudoti kuo daugiau skirtingų technologijų, taip demonstruojant sistemos universalumą, praktiškumą ir pakartotinį panaudojimą. Trūkstamų dalių realizacijai naudoti naujausias technologijas.

1.4. TYRIMO OBJEKTO ANALIZĖ

Pasaulinė rinkos analizės kompanija „Gartner“ korporacijos „Oracle“ produktą „Application Server 10g“ įvertino 2006-tų metų antrąjį pusmetį, paskirdama jam vietą lyderių ketvirtyje, o žurnalas „VARBusiness“ serverį pripažino metų produktu, skirtu vidutinio dydžio įmonėms. Mažoms ir vidutinėms įmonėms reikalinga kokybiška programinė įranga, kurią lengva įdiegti ir prižiūrėti. Be šių dviejų savybių yra nepaprastai svarbi dar viena- plėtimo galimybė, kuri reikalinga įmonei augant.[6][7]

Aukščiau minėti įvertinimai rodo, kad Oracle šias savybes tenkina geriausiai. Oracle Spatial yra Oracle DBVS komponentas, kuris suteikia galimybę naudotis efektyviais GIS ir LBS(Location based services) sprendimais ir yra rimtas pamatas sudėtingoms GIS taikomosioms programoms kurios reikalauja didelės erdvinės informacijos apdorojimo spartos. Oracle Spatial palaiko visus pagrindinius erdvinės informacijos tipus bei duomenų

modelius, realizuojant kritinius verslo sprendimus, tokius kaip viešas sektorius, gynybos departamentu, logistika ir kiti gyvenimo atvejai.

Oracle Spatial, suteikia SQL schemą ir funkcijas, kurios palengvina laikymą, paiešką, atnaujinimą, ir kolekcijų užklausas erdvinių ypatybių Oracle duomenų bazėje. Oracle Spatial sudaro[8][9]:

- schema (MDSYS), kuri nurodo laikymą, sintaksę, ir semantiką geometrinių duomenų, kuriuos palaiko Spatial
- erdvinis indeksavimo mechanizmas
- Operatoriai, funkcijos, ir procedūros, kurios reikalingos tam, kad įvykdytų dominamos srities užklausas, erdvines sujungimo užklausas ir kitas erdvines analizės operacijas
- Funkcijos ir procedūros universalumui ir operacijų derinimui
- Topologinių duomenų modelis tam, kad būtų galima dirbti su duomenimis apie mazgus, kampus, ir paviršių topologija
- Tinklo duomenų modelis kuris skirtas objektams, kurie yra modeliuoti kaip mazgai ir sąsajos, tinkle.
- GeoRaster - ypatybė, kuri leidžia kaupti, indeksuoti, daryti užklausas, analizuoti, ir tiekti GeoRaster duomenis, kurie yra iš taškų sudaryti paveiksliukai bei su jais susieti meta duomenys.

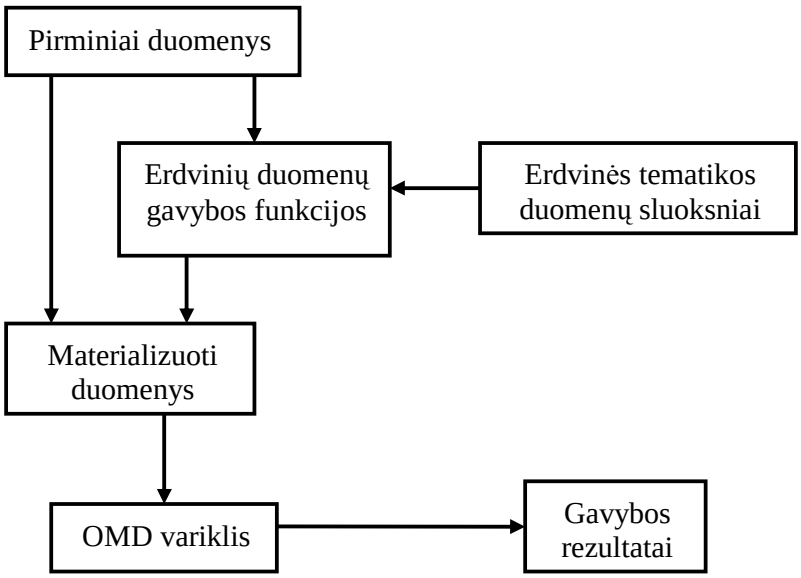
1.4.1. ERDVINIŲ DUOMENŲ GAVYBA (SPATIAL DATA MINING)

ODM (Oracle data mining) leidžia automatinį informacijos išgavimą iš duomenų bazės. Ši technologija apima paslėptų asociacijų tarp skirtingų duomenų atributų suradimą, duomenų klasifikavimą pagal pavyzdžius ir duomenų grupavimą, kad būtų galima sukurti susijusių duomenų vaizdą. Efektyvūs, su Oracle Duomenų baze susieti, erdviniai duomenys gali būti materializuoti įtraukimui į duomenų gavybos paraiškas.

Daugelyje taikomųjų programų, specifinės vietovės duomenys yra įtakojami duomenų susijusių su rajoną apibūdinančia informacija. Pavyzdžiui, namo vertė gali būti nustatoma pagal vertę kitų namų esančių tame rajone. Šį reiškinį vadiname erdvine koreliacija (ar rajono įtaka). Oracle Spatial erdvinė analizė ir duomenų gavybos (data mining) funkcionalumas leidžia eksploatuoti erdvinę koreliaciją vartojant vietovės požymius atitinkančius duomenų atributus keliais būdais: skirstant duomenis į sritis (tokių kaip šiaurines, pietų, rytų, ir vakarų sričių skirstymas kategorijomis), tam, kad būtų galima įvertinti kaimynystės objektų įtaką (kaip, pavyzdžiui, klientų skaičius kiekvieno prekybos centro aplinkinėje teritorijoje ir pan.), ir

tam, kad būtų galima identifikuoti šalia esančius konkurencinius objektus (tokius kaip video nuomos parduotuvės, picerijos, restoranai).

Kad būtų galima įvykdyti erdvinę duomenų gavybą, reikia suformuoti erdvinius predikatus ir ryšius erdvinių duomenų komplektui, naudojant teminius sluoksnius. Kiekvienas sluoksnis savyje turi duomenis apie specifinę erdvinių duomenų rūšį, pavyzdžiui, parkai ir poilsio sritys, ar demografiniai pajamų duomenys. Erdvinių duomenų materializavimas gali būti įvykdytas kaip išankstinio apdorojimo žingsnis prieš duomenų gavybos taikomosios programos vykdymą arba jis galėjo būti įvykdytas kaip tarpinis žingsnis erdvinėje duomenų gavyboje, kaip parodyta sekančiame paveiksle 2.[10]



2 pav. Erdvinių duomenų gavybos procesas[10]

- Pirminiai duomenys – tai yra duomenys kurie apima erdvinius ir neerdvinius duomenis ir yra apdirbami tam, kad iš jų būtų galima gauti materializuotus duomenis (duomenis kurie reiškia kažkokį realų objektą ant žemėlapio).
- Erdviniai duomenys esantys pirminiuose duomenyse yra apdirbami erdvinių duomenų gavybos funkcijų, kad iš jų būtų gauti materializuoti duomenys.
- ODM variklis apdoroja materializuotus duomenis (erdvinius ir neerdvinius), kad sukurtų duomenų gavybos rezultatus.

1.4.2. GEORASTER

GeoRaster yra Oracle Spatial ypatybė, kuri leidžia kaupti, indeksuoti, daryti užklausas, analizuoti ir tiekti GeoRaster duomenis, kurie yra, piešinukas ir pagal savo meta duomenis sugrupuoti į tinklelį. GeoRaster aprūpina Oracle Erdviniais duomenų tipais ir objektų ryšių schema. Šituos duomenų tipus, ir objektų ryšių schemas galima panaudoti, erdvinių duomenų ir įvairių sluoksnių kaupimui, kurie atitinka realius objektus esančius ant žemės paviršiaus, nurodytus vietinėje koordinačių sistemoje. Kiekvienas paveikslėlis ar multisluoksnio tinklelis yra sukaupti kaip GeoRaster objektai bet kurioje duomenų bazės lentelėje. Jei duomenys yra susieti su erdvine informacija, galima surasti vietą ant Žemės ląstelei rastre; ar atsižvelgiant į vietą ant Žemės, galima surasti ląstelę taškiniame sluoksnyje, susietame su ta vieta.

GeoRaster yra suprojektuotas, kad tiektų įmonėms paslaugas didelių vaizdų apdorojimui ir GIS sprendimams. Kūrėjams dabar galima sujungti šią galingą duomenų valdymo technologiją su pagrindiniu vaizdų apdirbimu ir rastro/grotelių analizės įrankiais.

GeoRaster architektūra suteikia, pagrindinį funkcionalumą kurio reikia norint suteikti vaizdo ar tinkleliais pagrįstų taškinių duomenų laikymą Oracle duomenų bazėje. Aukštame abstrakcijos lygyje, GeoRaster architektūrą sudaro šeši pagrindiniai komponentai[12]:

1. GeoRaster Variklis – Aprūpina GeoRaster objekto tipą ir pagrindinį GeoRaster funkcionalumą, įtraukdamas taškinius duomenis ir meta duomenų operacijas bei indeksavimą.

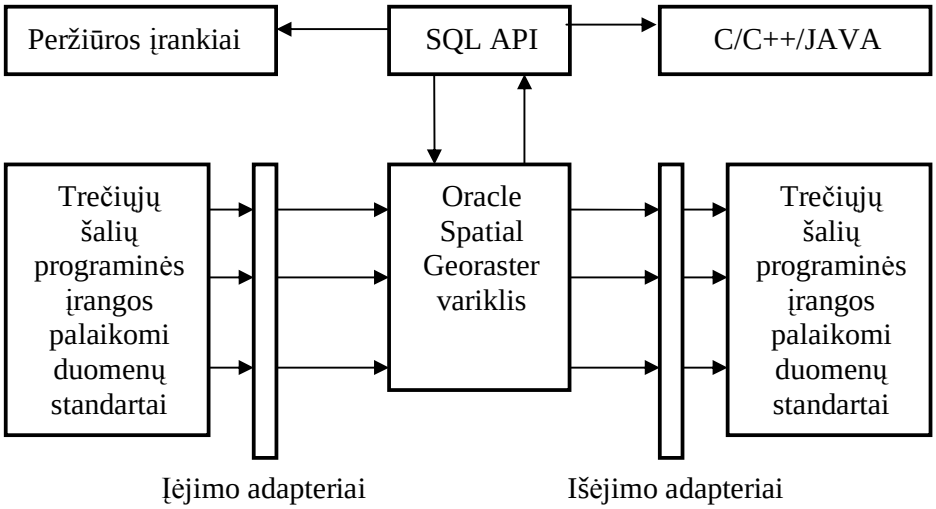
2. SQL programavimo sąsaja – SQL prieiga prie rastro ir tinkleliu pagrįstų duomenų GeoRaster.

3. C/C ++/Java – OCI, OCCI, ir Java prieiga prie duomenų ir rastro tinkleliu grindžiamų duomenų, kviečiant ar nekviečiant GeoRaster programavimo sąsajos.

4. Peržiūrėjimo Įrankiai: dauguma trečiosios šalies, peržiūros ir analizės įrankių dabar, palaiko GeoRaster. Oracle MapViewer palaiko GeoRaster.

5. Įvesties [duomenų] adapteriai – Palengvina gerai žinomo formato duomenų užkrovimą į GeoRaster.

6. Išėjimo [duomenų] adapteriai – Palengvina rastrinių duomenų iš GeoRaster išpakavimą į gerai žinomus formatus.



3 pav. GeoRaster architektūra[12]

Oracle GeoRaster branduolys yra fizinė architektūra rastro ar tinkleliais pagrįstų duomenų kaupimui ir valdymui duomenų bazėje. GeoRaster variklyje, toks duomenų tipas, pavadintas SDO_GEORASTER ir jame kiekvienas paveikslėlis, ar taškiniai tinkleliai yra saugomi kaip vienas šio domenų tipo objektas. GeoRaster duomenų lentelė yra bet kokia vartotojo apibrėžta lentelė, kuri turi mažiausiai vieną SDO_GEORASTER tipo duomenų skiltį.

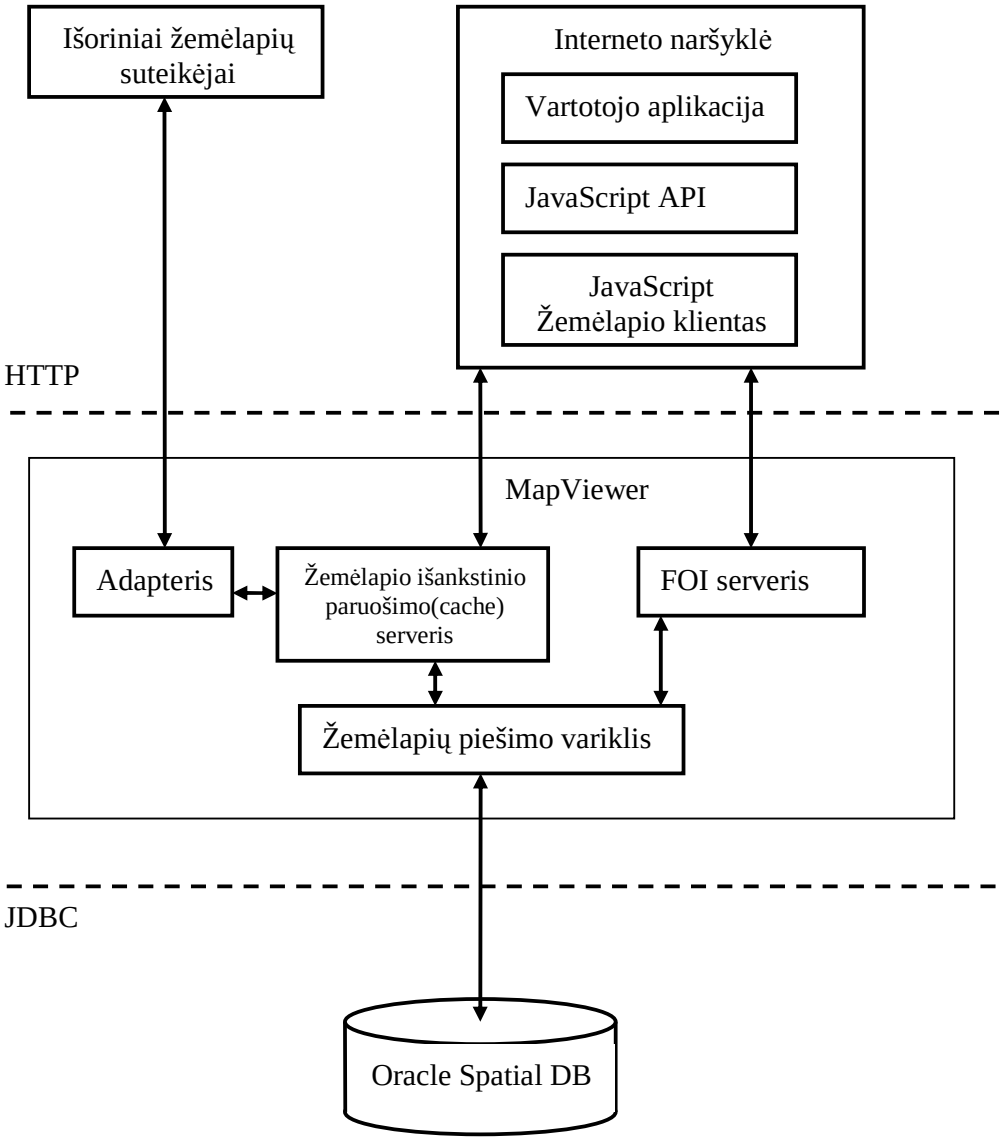
1.4.3. MAPVIEWER

Taikomoji programinė įranga Oracle mapViewer (OracleAS MapViewer) yra programuojamas komponentas kuris skirtas tam, kad generuotų žemėlapius, naudodamas erdvinius duomenis, valdomus Oracle Spatial ar Oracle Locator (taip pat vadinamo tiesiog Locator). OracleAS MapViewer suteikia įrankius, kurie paslepia erdvinių duomenų užklausų ir žemėlapio generavimo sunkumą, suteikdami įvairių pasirinkimų galimybę labiau pažangiems vartotojams. Šitie įrankiai gali būti išdėstyti nepriklausomu nuo platformos būdu ir yra suprojektuoti taip, kad būtų integruojami su žemėlapį teikiančiomis taikomosiomis programomis.

OracleAS MapViewer yra traktuojamas kaip Taikomojo Oracle Serverio dalis. Jo svarbiausia dalis yra J2EE taikomoji programa, kuri gali būti įdiegta į J2EE konteinerį. OracleAS MapViewer sudaro tokie svarbiausi komponentai[11]:

- pagrindinis piešimo variklis (Javos biblioteka) pavadintas SDOVIS, kuris įvykdo kartografinių vaizdų perteikimą. Tuo tikslu yra sukurtas Servlet‘as, kuris suteikia piešimo funkcijas internetinėms taikomosioms programoms.
- rinkinys taikomojo programavimo sąsajų (API), kurios leidžia programuojamą prieigą prie MapViewer ypatybių. Šis API apima XML, Java, PL/SQL, ir AJAX-pagrįstą JavaScript programavimo sąsają.
- grafinis žemėlapio kūrimo įrankis, kuris leidžia sukurti žemėlapio simbolius, apibrėžti erdvinius duomenis verčiančias taisykles ir sukurkti bei redaguoti MapViewer objektus.
- Oracle žemėlapis, kuris apima žemėlapių išankstinį paruošimą ir FOI serverius, kurie palengvina erdvine informacija besinaudojančių taikomųjų programų integravimą.

Žemiau pateiktame paveiksle 4 matome kaip MapViewer gali būti naudojamas vartotojo taikomųjų programų. MapViewer yra pasiekiamas iš kliento interneto naršyklių per HTTP protokolą. Vartotojui reikia turėti naršyklę kuri palaiko JavaScript, bei JavaScript žemėlapio kliento API. Kreipdamiesi klientai į MapViewer programinę įrangą turinčius serverius per žemėlapių piešimo variklį gali gauti norimus žemėlapius. Bendravimas tarp Oracle spatial duomenų bazės ir žemėlapių piešimo variklio yra realizuotas per JDBC tvarkyklę. Taip pat yra galimybė gauti išorinių tiekėjų duodamus žemėlapius per adapterį.



4 pav. MapViewer architektūra[11]

Kai kokia nors taikomoji programa kreipiasi į OracleAS MapViewer, jis taiko specifinius stilius (tokius kaip spalvos ir struktūra) ir specifines temas (kurios yra, kolekcijos erdvinių ypatybių, tokios kaip miestai, upės, ir magistralės) kad suteiktų tinkamą žemėlapį (tokį kaip GIF vaizdas parodymui ant tinklalapio). Pavyzdžiui, taikomoji programa galėtų rodyti žemėlapį, kuriame valstybiniai parkai pasirodo nuspalvinti žaliai, o restoranai yra pažymėti raudonomis žvaigždėmis. Žemėlapis tipiškai turi kelias temas, atstovaujančias politiniam ar fiziniam objektui, ar abiem. Pavyzdžiui, žemėlapis galėtų parodyti nacionalines ir valstybines sienas, miestus, kalnynus, upes, ir istorines vietas. Kai žemėlapis yra sugeneruojamas, kiekviena tema atitinka tam tikrą sluoksniui užbaigtame paveikslėlyje.

1.5. VARTOTOJŲ ANALIZĖ

1.5.1. VARTOTOJŲ AIBĖ, TIPAI IR SAVYBĖS

Projektuojamos geografinės informacinės sistemos vartotojais galėtų būti:

- Smulkaus ir vidutinio verslo įmonės – tai įmonės kurios naudotų šią paslaugą verslo tikslais. Paslaugos būtų skirtos klientams(komercines paslaugos) arba vidiniams darbuotojams (įmonės plėtros ar analizės procesui palengvinti)
- Viešosios įstaigos – tai valstybinės įstaigos kurios naudotųsi šia paslauga švietimo ar informavimo tikslais.

1.5.2. VARTOTOJŲ TIKSLAI IR PROBLEMOS

Projektuojama, verslo geografinė informacinė sistema padės įvairių verslo sričių specialistams susipažinti ir panaudoti GIS savo asmeninio verslo plėtrai bei konkurencingumui didinti.

- Smulkaus ir vidutinio verslo įmonės – naudojasi GIS sistema tam, kad susietų savo turimus specifinius duomenis su erdvine informacija ir pateiktų vartotojui vaizdžioje formoje. Tokiu būdų palengvintų savo verslo procesus, padarytų juos labiau suprantamus vartotojams bei padidintų jų efektyvumą. Vidiniai įmonės darbuotojai naudodamiesi informacija pateikta žemėlapyje galėtų greičiau ir aiškiau planuoti įmonės plėtros galimybes ir pan.
- Viešosios įstaigos – pavyzdžiui mokyklos, greitoji pagalba, policija ir t.t. gali naudoti šią sistemą ne verslo, tačiau švietimo, analizės ar informavimo tikslais. Taipogi tokia sistema galėtų padėti greitai suvokti iškilusias skubias problemas ir į jas reaguoti.

1.6. ESAMŲ SPRENDIMŲ ANALIZĖ

Kadangi šiuo metu GIS sistemų kūrimui ir palaikymui yra skiriama daug dėmesio, todėl ir egzistuojančių sprendimų yra tikrai daug. Šiame darbe yra tiriamos Oracle DBVS teikiamos GIS galimybės, todėl analizuojant panašius sprendimus labiau orientuosimės į sprendimus kuriuose erdvinės informacijos saugojimui ir apdorojimui naudojama Oracle Spatial teikiamos galimybės. Analizuojant panašius sprendimus yra tikslinga atlikti panašių sprendimų analizę Lietuvos ir tarptautiniu mastu. Analizei pateikiami sprendimai:

- GEOLIS
- Verslo GIS
- Oracle – Autodesk sprendimas
- Google maps API

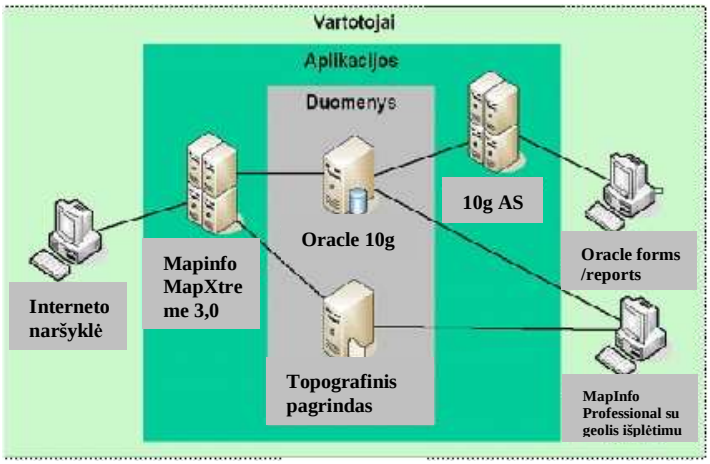
1.6.1. GEOLIS

GEOLIS – tai integruotų informacinių technologijų infrastruktūra, tarnaujanti duomenų apie Žemės gelmių išteklius bei kitos geologinės informacijos kaupimui, apdorojimui ir pateikimui vartotojams reikalinga forma. Lietuvos geologijos tarnybos (LGT) informacinė sistema yra galingas geologinės informacijos tvarkymo įrankis, kuriuo naudojasi gamtosaugos, mokslinės, inžinerinės, gavybos, tyrimų bei gavybos įmonės ir organizacijos. Šis infrastruktūrinis sprendimas apjungia įvairiapusę įvairių organizacijų (valstybinių institucijų, mokslo įstaigų, privačių kompanijų) veiklą į vieningą LGT koordinuojamą geologinės informacijos apykaitos procesą.[13]

Geologinės informacijos sistema yra sukurta Oracle ir MapInfo korporacijų programinės įrangos pagrindu. Centrinė duomenų bazė ir neerdvinės informacijos valdymo aplinkos šiuo metu veikia Oracle 10g (su Oracle Spatial) bei Oracle Application Server (AS) 10g pagrindu.

Pagrindinis erdvinės informacijos kūrimo ir valdymo įrankis yra MapInfo Professional 7.5 naudojamas erdvinių duomenų kūrimui, Oracle Spatial erdvinės informacijos valdymui, geologinių žemėlapių ir kitos kartografinės medžiagos kūrimui ir paruošimui, leidybai, erdvinių duomenų apsikeitimui su kitomis GIS sistemomis, reikalingos geologinės informacijos paieškai ir analizei ir kt.

Erdvinės informacijos pateikimui Internete įdiegta MapInfo MapXtreme 3.0 programinė įranga. Ši programinė įranga leidžia įvairius geologinius duomenis pateikti Interneto vartotojams skaitmeninio žemėlapio forma.[13]



5 pav. GEOLIS architektūra [13]

Svarbu atkreipti dėmesį į tai, kad šioje sistemoje yra skiriamas dėmesys erdvinės ir neerdvinės informacijos susiejimui. Tuo tikslu MapInfo Professional pagrindu yra sukurtas GIS - Geolis programinis modulis, kuris leidžia skaitmeninio žemėlapio aplinkoje greitai ir patogiai naudoti centrinėje Oracle duomenų bazėje saugomą informaciją. GEOLIS yra sukurtas integruoto duomenų modelio pagrindu. Tai leido pasiekti bene didžiausią duomenų integralumą Lietuvoje. Šiuo metu Interneto vartotojams skaitmeninio žemėlapio forma pateikiama labai daug informacijos apie Lietuvos teritorijos geologinę sąrangą bei žemės gelmių išteklius. Didelė dalis šios informacijos pasiekiama plačiajai visuomenei.

1.6.2. VERSLO GIS

Nepaisant to, kad panašių sprendimų analizei didesnį dėmesį skyrėme sistemoms kurios yra sukurtos naudojant Oracle DBVS, yra tikslinga atlikti jau egzistuojančių GIS paslaugų skirtų mažoms ir vidutinėms verslo įmonėms, kurios nenaudoja Oracle DBVS teikiamų galimybių. Natūralu, kad daugelis GIS paslaugų gali būti realizuotos naudojant skirtingas technologijas, todėl yra svarbu palyginti jas su Oracle Spatial teikiamomis galimybėmis, suprasti skirtingų technologijų privalumus ir trūkumus. Tuo tikslu yra tikslinga išanalizuoti „Verslo GIS“ nemokamą e-paslaugą.

Projektas "Verslo geografinė informacinė sistema", įgyvendintas Phare 2002 programos lėšomis, yra skirtas skatinti verslo plėtrą Kauno mieste, pagerinti viešųjų paslaugų verslui kokybę, įvairovę ir prieinamumą. Projektas įgyvendintas taikant pažangias internetines GIS technologijas. Verslo GIS teikia galimybę nemokamai naudotis įrankiu bei duomenimis, skirtais geografiniam verslo aplinkos, poreikio ir perspektyvumo įvertinimui. Sukurta sistema leidžia atlikti interaktyvią geografinę informacijos paiešką ir analizę integruojant įvairius duomenis iš skirtingų šaltinių:[14]

- SĮ “Kauno planas” teikiamą Kauno miesto georeferencinį pagrindą;
- UAB “ENIRO” įmonių katalogo duomenis;
- UAB “InfoBankas” įmonių kreditingumo informaciją;
- Statistikos departamento teikiamus Gyventojų surašymo duomenis.

Tikslinės verslo geografinės informacinės sistemos naudotojų grupės yra šios:

- Smulkaus ir vidutinio verslo įmonės (naudojasi paslaugomis pačios);
- Verslo asociacijos ir konsultacinės kompanijos (paslaugomis naudojasi analizei ir kitų verslo subjektų konsultavimui);
- Kauno miesto savivaldybė (verslo analizei ir planavimui).

Paslaugos paskirtis – informacijos apie verslo įmones paieška pagal dalykinį paieškos kriterijų laisvai pasirenkamoje Kauno miesto teritorijoje, taip pat įmonių ekonominės veiklos apibendrintos informacijos pateikimas miesto kvartalų arba miesto dalių ar seniūnijų teritorijų lygiu. Apibendrinta informacija interaktyviai analizuojama kartu su turima tų miesto teritorijų statistika, skaičiuojamos išvestinės charakteristikos, o rezultatai vaizduojami tematiniais žemėlapiais.

Technologinis sistemos funkcionalumas užtikrinamas atvirojo kodo programine įranga – MapServer žemėlapių serveriu, MySQL duomenų bazių valdymo sistema, PHP programavimo kalba. Funkcinės sistemos dalies projektavimo ir programavimo darbus atliko UAB „Informacinės technologijos“ geoinformacinių sistemų skyriaus specialistai[14].

1.6.3. AUTODESK IR ORACLE GIS SPRENDIMAS

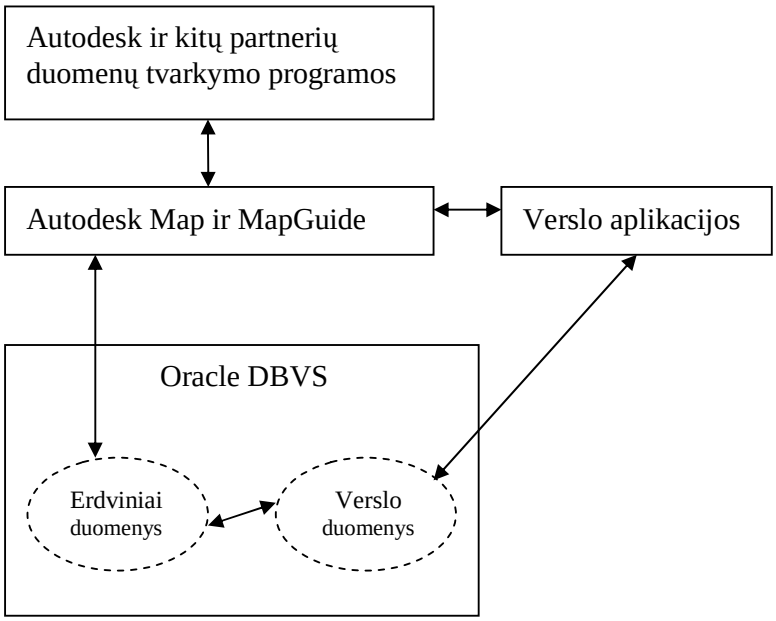
Ieškant panašių sprendimų tarptautiniu mastu, jų galima aptikti tikrai daug, kadangi efektyvių GIS sistemų aktualumas yra didelis, todėl ir spendimų pasiūla nemenka. Rinkoje pagrindinis vaidmuo atitenka ESRI kompanijai kuri yra sukūrusi daug programinės įrangos skirtos erdvinės informacijos saugojimui apdorojimui ir vaizdavimui. Oracle savo ruožtu taip pat užima vis didesnę dalį GIS sistemų rinkoje. Pagrindinė Oracle DBVS funkcija GIS sistemų kūrime yra erdvinių duomenų saugojimas ir greitas jų apdorojimas. Žemėlapių ir objektų ant jų piešimui paprastai naudojama papildoma programinė įranga, leidžianti vartotojui naudoti žemėlapius tiesiog savo taikomojoje programoje. Plačiau panagrinėsime Autodesk kompanijos siūlomą sprendimą žemėlapiams iš Oracle Spatial išgauti.

Naudojant Autodesk integruotomis susiejimo su Oracle galimybėmis, galima valdyti didelius erdvinės informacijos kiekius atviroje informacijos terpėje. Įmonės kurių veikla

priklauso nuo erdvinių duomenų gali sujungti Autodesk erdvinius sprendimus su Oracle duomenų bazėje esančia erdvine informacija, tokiu būdu sumažinant išlaidas ir padidinant funkcionalumą.

Šis galingas erdvinės informacijos vizualizavimo įrankis leidžia vartotojams tvarkyti, analizuoti bei peržiūrėti su vietove susijusią informaciją greitai efektyviai ir pigiai. Kadangi Autodesk sprendimai integruojami tiesiogiai su Oracle Spatial įmonėms nebereikia jokios tarpinės programinės įrangos kritinių verslo duomenų pasiekimui. Erdviniai duomenys gali būti tvarkomi naudojant Oracle teikiamus įrankius, nenaudojant jokios tarpinės GIS programinės įrangos. Duomenis iš Oracle duomenų bazės gali būti paimami per SQL užklausas naudojant standartines tvarkykles (ODBC, JDBC, OLE DB). Tai vėlgi nereikalauja jokių papildomų tarpinių GIS platformų.

Dėl atviros savo architektūros Oracle – Autodesk sprendimas leidžia valdyti erdvinę informaciją naudojantis Oracle įrankiais, o tai suteikia daugiau lankstumo ir paprastumo duomenų administratoriams. Erdvinė informacija Oracle duomenų bazėje yra saugoma tokia pat lentelių struktūra kaip ir dauguma kitų, mums įprastų, duomenų lentelių.[17]



6 pav. Oracle – Autodesk sprendimas [17]

Aukščiau pateiktame paveiksle 6 pavaizduotas Oracle – Autodesk sprendimas. Kaip matome šiuo atveju verslo ir erdviniai duomenys yra saugomi naudojant tą pačią DBVS. Tokiu būdu galima susieti šiuos duomenis vidinėmis užklausomis arba per vartotojo

taikomąją programą. Autodesk Map ir MapGuide yra Autodesk sukurta programinė įranga, kuri suteikia galimybę piešti žemėlapius bei kurti erdvinę informacija bei ją apdoroti.

1.6.4. GOOGLE MAPS API

Google Žemėlapių programavimo sąsaja (API) leidžia naudoti Google žemėlapius savo tinklalapiuose. JavaScript. Programavimo sąsaja leidžia naudotis daugeliu korporacijos Google teikimų funkcijų, kurios skirtos generuoti žemėlapius, naudojant erdvinę ir aprašomąją informaciją saugomą Google serveriuose.

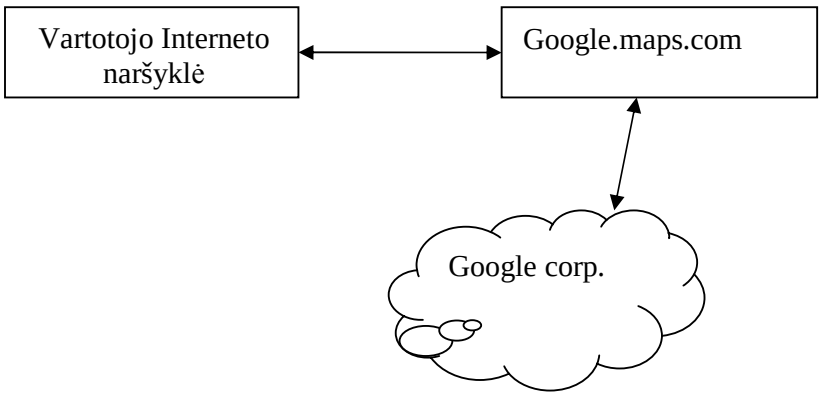
Google Žemėlapių programavimo sąsaja yra pagrįsta JavaScript, kuri suteikia lanksčias funkcijas generuojant žemėlapius. Naudojantis šiuo API nereikia parsisiųsti jokios papildomos programinės įrangos ir galima visiškai nesigilinti kaip žemėlapis yra generuojamas, kadangi visa tai atlieka google realizuotos funkcijos. Reikia tik išstudijuoti šio API teikiamas funkcijas ir bus galima laisvai programuoti žemėlapių gavimą pagal savo poreikius, naudojantis google teikiamais erdviniais ir aprašomaisiais duomenimis[15].

Google žemėlapiai yra suderinti su visomis naršyklėmis kurios palaiko javaScript programavimo kalbą.

Pagrindinis GoogleMaps privalumas yra tas, kad korporacija yra sukaupus didelius kiekius erdvinių duomenų, o aprašomuosius duomenis nuolatos atnaujinti gali patys vartotojai, todėl naudojantis šiuo API programuotojai gali generuoti žemėlapius ir suprogramuoti programas kurios analizuotų verslo duomenis ir vaizduotų juos žemėlapyje.

Google Maps API suteikia tokias galimybes žemėlapių naudotojams:

- Geokodavimas
- Žemėlapiai gali sugeneruoti bet kokiam tinklalapiui
- Realizuota JavaScript pagalba
- Ajax galimybės žemėlapių ir informacijos pateikimo ant jų valdymui
- Lankstus žemėlapių valdymas pelės pagalba.
- Galima pridėti XML, XSL ar HTML tekstą ir vaizduoti objektus ant žemėlių
- Lankstus panaudojimas netgi mobiliame telefone



7 pav. Google žemėlapių panaudojimas

Paveikslėlyje 7 parodyta kaip gali būti panaudoti Google žemėlapiai. Interneto naršyklėje reikia nurodyti adresą kuriuo nurodoma JavaScript‘u komponento alokacija nutolusiame google serveryje. Naudojantis tokia architektūra, nereikia parsisiųsti jokių API žemėlapių valdymui. Viskas ko reikia tai tik raktas kuris leidžia naudoti šiuos nutolusiame kompiuteryje esančius komponentus. Likusi dalis (žemėlapių gavimas, apdorojimas ir t.t.) vartotojui ar netgi taikomųjų programų kūrėjui lieka visiškai nematoma. Tai yra komercinė Google korporacijos paslaptis. Vartotojui pakanka tik galinio rezultato ir visiškai nėra svarbu kokiu būdu jis buvo gautas.

1.6.5. ESAMŲ SPRENDIMŲ ANALIZĖS REZULTATAI

Išanalizavus panašius sprendimus, yra tikslinga palyginti juos tarpusavyje. Tokiu būdu galima įvertinti kuo viena sistema yra geresnė už kitą ir pastebėti kokio funkcionalumo trūksta kelioms arba visoms analizuotoms sistemoms. Analizuotose sistemose galima pastebėti dviejų tipų sistemas. T.y. sistemas kurios naudoja sukauptus erdvinius ir/arba aprašomuosius duomenis tik tam, kad pateiktų žemėlapį vartotojo analizei, bet neleidžia (nesuteikia jokio API) vartotojams patiems vaizduoti savo aprašomuosius duomenis žemėlapyje (pvz. verslo GIS). Sekančio tipo sistemos suteikia API klientų taikomosioms programoms (pvz.: google maps). Tam, kad jas visas palygintume sudarysime lentelę ir įvesime kriterijus pagal kuriuos vertinsime sistemas tarpusavyje. Jeigu sistema neturi tokio funkcionalumo kuris atitinka iškeltą kriterijų , tai atitinkamame laukelyje žymėsime „nėra“, jei funkcionalumas yra, tačiau juo sudėtinga naudotis, žymėsime „-“. Jei funkcionalumas yra ir juo paprasta ir aišku pasinaudoti, žymėsime „+“. Žemiau aprašomi kriterijai pagal kuriuos buvo vertintos sistemos:

- Sprendimo naujumas – šis kriterijus nurodo ar sistemos architektūra ir/arba veikimo principas yra unikalus ir naujas.
- Pakartotinas panaudojimas – nusako ar sistemos architektūros dalis arba API gali būti pakartotinai panaudojami kitose taikomosiose programose neribotą kartų kiekį.
- Panaudojimo paprastumas – nusako ar sąlyginai nedaug programavimo, konfigūravimo ar integravimo darbų reikia atlikti norint vartotojų taikomosiose programos naudoti skaitmeninius žemėlapius.
- Konfigūravimo paprastumas – jei GIS sistemos architektūroje yra vartotojo laisvai konfigūruojamas komponentas, tai šis kriterijus nusako, ar konfigūravimui atlikti reikia minimalių pastangų.
- Architektūros išskaidymas – nusako ar GIS sistemos atskiros dalys gali būti talpinamos atskiruose nutolusiuose kompiuteriuose. Tokiu būdu viena nuo kitos yra atskiriamos skirtingos sistemos dalių funkcijos tam, kad jas būtų galima lengviau prižiūrėti ir palaikyti.
- Esamų duomenų panaudojimas – galimybė į GIS sistemą importuoti jau egzistuojančius erdvinius duomenis.
- Naujų duomenų kūrimas – galimybė GIS sistemą papildyti naujais erdviniais duomenimis.
- Nepriklausomumas nuo DB – galimybė automatiškai užkrauti įmonės aprašomuosius duomenis ant skaitmeninių žemėlapių nepriklausomai nuo duomenų bazės kurioje jie saugomi tipo.
- Aprašomųjų duomenų papildymas- jei sistema leidžia automatizuotu būdu vaizduoti aprašomuosius duomenis žemėlapyje, tai šis kriterijus nurodo ar norint šalia esančių aprašomųjų duomenų pridėti naujus, reikia minimalių pastangų.
- Naujų vaizdavimo galimybių įtraukimas - jei sistema leidžia automatizuotu būdu vaizduoti aprašomuosius duomenis žemėlapyje, tai šis kriterijus parodo ar daug pastangų reikės ateityje iš kūrėjų pusės siekiant įdėti naujų žemėlapių vaizdavimo būdų ir ar tai turės minimalias pasekmes klientų pusėje.
- Dinamiškumas – rodo ar naudojantis GIS sistema galima pasiekti dinamiškų rezultatų t.y. ar viena kartą sukompiliavus ir sukonfigūravus GIS sistemą į klientų poreikių pasikeitimus bus galima reaguoti dinamiškai, nekeičiant programos kodo.
- Rezultatų aiškumas – rodo ar aiškiai žemėlapyje matoma klientų pageidaujama informacija.

- Programavimo darbų minimizavimas – parodo ar siekiant suprogramuoti vartotojo taikomąją programą kuri naudotų skaitmeninius žemėlapius, reikia atlikti minimalius programavimo darbus kurie susiję su informacijos atvaizdavimu žemėlapyje ir žemėlapio įdėjimu į taikomąją programą.
- Investicijų mažinimas – parodo ar naudojantis GIS sistema įmonėms reikės minimalių investicijų žemėlapiams ir ar ateityje reikės minimalių investicijų taikomųjų programų palaikymui.

1 lentelė. Esamų sprendimų palyginimas

	GEOLIS	**VERSLO GIS**	**ORACLE AUTODESK**	**GOOGLE MAPS**
Sprendimo naujumas	-	-	-	-
Pakartotinas panaudojimas	-+	-+	+	+
Panaudojimo paprastumas	-+	-+	+	+
Konfigūravimo paprastumas	nėra	nėra	-+	-+
Architektūros išskaidymas	+	+	+	+
Esamų duomenų panaudojimas	+	+	+	+
Naujų duomenų kūrimas	+	+	+	+
Nepriklausomumas nuo DB	nėra	nėra	nėra	nėra
Nepriklausomumas nuo verslo duomenų	nėra	nėra	nėra	nėra
Naujų vaizdavimo galimybių įtraukimas	-	-	-	-+
Dinamiškumas	-	-	-	-+
Rezultatų aiškumas	-+	+	-+	+
Programavimo darbų minimizavimas	-	-	+	-+
Investicijų mažinimas	-	-	-+	-+

1.7. SIEKIAMAS SPRENDIMAS

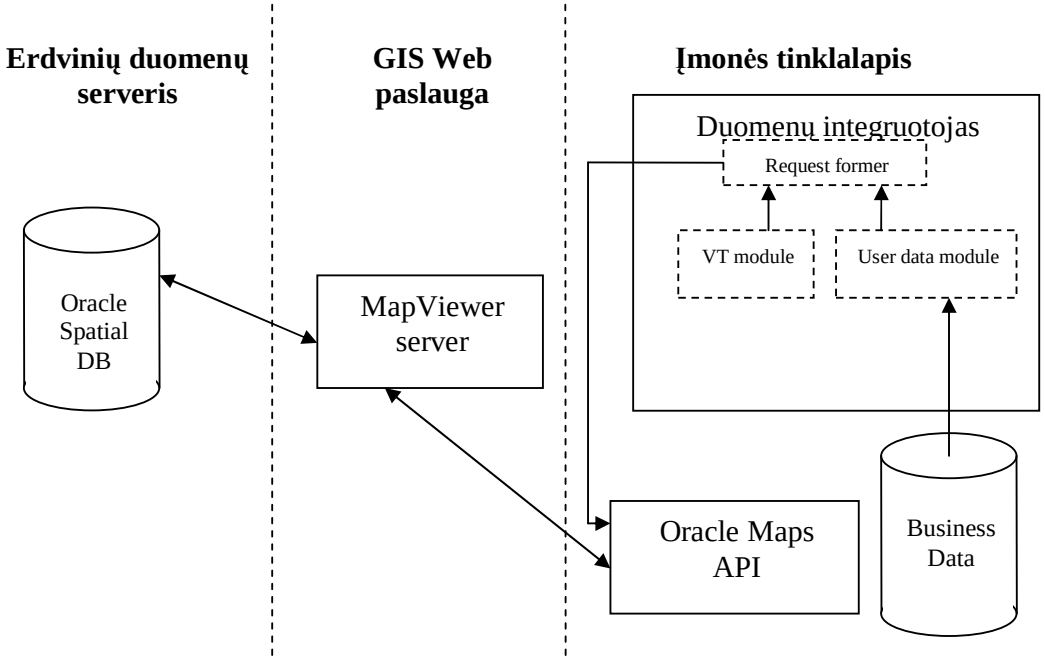
Kuriamas komponentas turės palengvinti verslo duomenų ir erdvinės informacijos integravimą nepriklausomai nuo verslo duomenų struktūros ir saugojimo vietos, kadangi skirtingos verslo šakos turi skirtingų tipų duomenis, kurie gali būti saugomi daugelyje skirtingose pasaulio vietose esančiuose serveriuose. Tokiu būdu naudojantis tuo pačiu komponentu, skirtingo tipo verslo duomenys iš skirtingų vietų galės būti susiejami su erdvine informacija esančia vieningoje Oracle duomenų bazėje.

Kadangi didelė dalis erdvinės informacijos jau egzistuoja, todėl reikės rasti būdą, kurio pagalba šią informaciją būtų galima importuoti į Oracle duomenų bazę. Mažoms ir vidutinėms įmonėms nebus būtina turėti šių duomenų kopijų savo žinioje, pakaks vienos ar keleto duomenų bazių prie kurių komponentas galės prisijungti nuotoliniu būdu ir gauti reikiamus duomenis bei susieti juos su verslo duomenimis.

Mažos ir vidutinio dydžio įmonės nori naudotis optimaliais GIS sprendimais už mažiausią kainą. Dėl šios priežasties kiekviena įmonė negali sau leisti įsigyti nuosavų skaitmeninių žemėlapių ir susipirkti visą programinę įrangą reikalingą jų administravimui bei apdorojimui. Dėl anksčiu išvardintų priežasčių tikslinga sukurti vieną erdvinės informacijos saugyklą, kurioje būtų talpinama erdvinė informacija ir šią informaciją galėtų naudoti daugelis įmonių. Taip pat yra būtina maksimaliai sumažinti programinės įrangos kiekį, kurį turi įsidiegti įmonė tam, kad savo taikomosiose programose galėtų naudotis GIS paslaugomis. Norint įgyvendinti šiuos tikslus, reikia sukurti internetinę paslaugą, kuri sugebės aptarnauti įmonių poreikius nuotoliniu būdu. Tuo tikslu turi egzistuoti serveris kuriame yra saugoma erdvinė informacija. Visa programinė įranga šios informacijos valdymui turi būti įdiegta internetinės paslaugos dalyje, tokiu būdu įmonės yra išlaisvinamos nuo papildomo programinės įrangos diegimo.

Kadangi daugelis įmonių žemėlapius nori vaizduoti internetiniuose puslapiuose, reikia sukurti komponentą kurio pagalba paveikslėlis kuriame yra sugeneruotas žemėlapis būtų lengvai užkraunamas į įmonių internetinius puslapius. Taip pat reikia realizuoti metodus, kurie galėtų susieti verslo duomenis su erdviniais duomenimis nereikalaujant didelių vartotojo pastangų. T.y. vartotojas nežino ir neturi nieko bendra su tuo, kokiu būdu ir kokiam pavidale yra saugomi erdviniai duomenys. Jis žino tik tai tą, ką jis nori matyti. Šiam tikslui pasiekti, galima iš anksto parengti taisyklių šablonus, kurių dėka verslo duomenys iš daugelių nutolusių duomenų bazių būtų susiejami su erdvine informacija. Veiklos taisyklių dėka, įmonėms siekiant susieti savo aprašomuosius duomenis su erdviniais duomenimis, nereikės atlikti programavimo darbų, pakas veiklos taisyklių pagalba aprašyti kokius jam

priklausančius duomenis jis nori matyti žemėlapyje. Tam, kad pagal užduotas veiklos taisykles sėkmingai vyktų erdvinių ir aprašomųjų duomenų integracija, pakas vieną kartą tinkamai sukonfigūruoti duomenų integravimo komponentą nurodant reikiamus prisijungimus prie duomenų šaltinių ir aprašant jų metaduomenis.



8 pav. Siekiamas sprendimas

8 pav. matome grafiškai pavaizduotą siekiamą sprendimą. Aiškiai galima atskirti tris dalis: erdvinių duomenų serverį, GIS web paslaugą ir įmonės tinklalapį. Akivaizdu, jog erdviniai duomenys ir jų apdorojimas yra atskirti nuo įmonės tinklalapio. Įmonės tinklalapis turi reikiamas sąsajas komunikavimui su GIS web paslauga, kad galėtų įmonės tinklalapį aprūpinti žemėlapiais pagal veiklos taisyklėmis aprašytus aprašomuosius duomenis. Veiklos taisyklės užduodamos „Duomenų integruotojo“ komponente, kuris jas sukompiliuoja ir vykdo.

1.8.ANALIZĖS IŠVADOS

Geografinės informacinės sistemos yra būtinos šiuolaikiniame versle, kadangi įmonėms reikia greitai ir efektyviai operuoti erdvine informacija integruojant ją su verslo sprendimais. Šiuo metu egzistuoja daugybė GIS sistemų ir GIS paslaugas teikiančios programinės įrangos. Tačiau apžvelgus skirtingo pobūdžio geografines informacines sistemas, galima pateikti jų trūkumus ir kryptis, kuriose yra padaryta labai nedaug:

- Dauguma esamų verslo duomenų ir erdvinės informacijos integravimo sprendimų nėra universalūs. Problema iškyla, nes integruojami verslo ir erdviniai duomenys jau iš anksto yra žinomi ir dažniausiai saugomi toje pat duomenų bazėje(GEOLIS atveju). Vartotojui yra pateikiamas tik integravimo rezultatas, t.y. vartotojas gali naudotis tik tais duomenimis kurie iš anksto yra žinomi sistemai kuria jis naudojasi (gatvių pavadinimai, regionai, įmonių adresai ir t.t.). Tokiu būdu vartotojas galbūt gali rasti informaciją kuri jį domina, tačiau negali atnaujinti šių duomenų. Kitaip tariant tokiose sistemose verslo ir erdviniai duomenys jau yra integruoti ir turi būti nuolatos atnaujinami. Tokios sistemos yra labai ribotos ir nesuteikia jokio API kuris leistų vartotojams naudotis jų sukauptais duomenimis.
- Norint operuoti erdviniais duomenimis, reikia įsidiegti visą eilę brangios programinės įrangos. Tai yra nepriimtina mažoms ir vidutinėms verslo įmonėms. Yra tikslinga šią programinę įrangą įdiegti išorinių paslaugų serverio dalyje, tokiu būdu įmonės išlaisvinamos nuo papildomų investicijų.
- Egzistuojantys žemėlapių generavimo komponentai yra efektyvūs, tačiau pertekliniai ir per brangūs siekiant optimalaus sprendimo verslui. Žemėlapio įkėlimui į internetinį puslapį reikia naudoti kiek įmanoma paprastesnį API.
- Žemėlapių gigantas Google plačiai prieinamam žemėlapių naudojimui pateikia tikrai lankstų sprendimą, tačiau ir jis yra ribotas. Google laisvam naudojimui atiduoda ištisus terabaitus sukauptos erdvinės informacijos, tačiau apribojimai atsiranda toje vietoje, kad trečioji šalis turi pati suprogramuoti kodo dalį kuri realizuoja ką, kur ir kokioje formoje norima matyti. T.y. verslo ir erdvinių duomenų integravimas yra statiškas.
- Geriausiai įmonių poreikius atitiktų toks sprendimas, kurio dėka erdviniai ir verslo duomenys būtų integruojami dinamiškai. T.y. programuotojui nereikėtų programuoti įmonės sukauptų duomenų perkėlimo ant žemėlapio logikos, tą galima atlikti automatiškai, todėl GIS sistemų panaudojimas versle taptų daug paprastesnis ir lankstesnis.

- Maksimalų panaudojimo efektyvumą atitiktų toks komponentas, kurio konfigūravimui siekiant gauti norimus rezultatus reikėtų kiek galima mažiau pastangų.
- Kuriamos sistemos naujumas yra tas, kad bet kokie verslo duomenys automatiškai, be programuotojo pagalbos galės būti atvaizduojami skaitmeniniame žemėlapyje, ko pasigendama visose analizuotose sistemose.
- Kuriamos sistemos konfigūravimo lankstumui pasiekti galima panaudoti veiklos taisykles, kurių pagalba galima paprasčiausiai natūralia kalba aprašyti kokius verslo duomenis reikia analizuoti ir kaip juos atvaizduoti žemėlapyje.
- Oracle DBVS turi tinkamą funkcionalumą ir komponentus lanksčių GIS sistemų karkaso kūrimui, kuris apima erdvinių duomenų saugojimą, jų išgavimą, bei žemėlapių generavimą. Dinamiškam verslo informacijos perkėlimui ant žemėlapio reikės sukurti naują komponentą, kuris pagal sukompiliuotas veiklos taisykles atvaizduos žemėlapyje atitinkamus duomenis.

2. REIKALAVIMAI DINAMIŠKAI GIS SISTEMAI

2.1.SISTEMOS VARTOTOJO SĄVOKA

Dinamiškos GIS sistemos paskirtis - palengvinti erdvinės informacijos panaudojimą kasdieniniuose verslo procesuose. Tai apima:

- Vidinių įmonės poreikių naudojant GIS sistemą tenkinimą
- Įmonių teikiamų paslaugų klientams pagerinimą.

Vidinių įmonės poreikių tenkinimas. Vis daugiau įmonių savo kasdieninėje veikloje susiduria su poreikiu greitai ir efektyviai išanalizuoti sukauptus savo veiklos duomenis ir matyti juos aiškioje bei patogioje formoje. Analizės rezultatus neretai reikia pateikti žemėlapyje bei atkreipti dėmesį į kritines sritis. Esant dideliam duomenų kiekiui ir įvairovei neretai tenka atlikti šių duomenų analizę keliais pjūviais. Kuo daugiau duomenų – tuo ilgiau trunka jų analizė bei rezultatų pateikimas. Norint išspręsti minėtas problemas, įmonės samdo programuotojus, kurie sukuria taikomąsias programas kurios paspartina duomenų analizę ir suteikia galimybę rezultatus matyti skaitmeniniuose žemėlapiuose. Šių taikomųjų programų sudėtingumas tiesiogiai proporcingas analizuojamų duomenų kiekiui ir pageidaujamam sistemų lankstumui. Neretai nutinka taip, kad sukuriamos sistemos puikiai atitinka pradinius reikalavimus, tačiau kai atsiranda poreikis sukurti naujas funkcijas arba analizuoti didesnį informacijos kiekį, tokias taikomąsias programas reikia perprogramuoti iš pagrindų. Tokiu būdu įmonės moka didelius pinigus savo naudojamų GIS sistemų palaikymui.

Vidinių įmonės poreikių pavyzdys: įsivaizduokime miškų gesinimo tarnybą, kuri turi surinkus daug duomenų apie visus gaisrus: kur ir kada gaisrai buvo. Norėdami operatyviai pasiruošti galimam gaisru pagausėjimui sezono metu(pvz.: vasarą), gaisrininkai turi žinoti kur ir kuriuo metu miškų gaisringumas buvo didžiausias. Akivaizdu, kad šiuo atveju jie galėtų pasinaudoti dinamiška GIS sistema suintegruodami turimus duomenis apie gaisrus su erdvine informacija (realiomis miškų koordinatėmis) saugoma GIS paslaugos serveryje ir tokiu būdu gauti žemėlapį kuriame matytų zonas kur miškų gaisringumas vasarą būna didžiausias.

Minėtai problemai spręsti dinamiška GIS tinka puikiai, kadangi programuotojams nereikėtų programuoti visos sistemos, tereiktų tik nurodyti verslo duomenis ir parametrus, pagal kuriuos norime atlikti verslo ir erdvinių duomenų integraciją į žemėlapį.

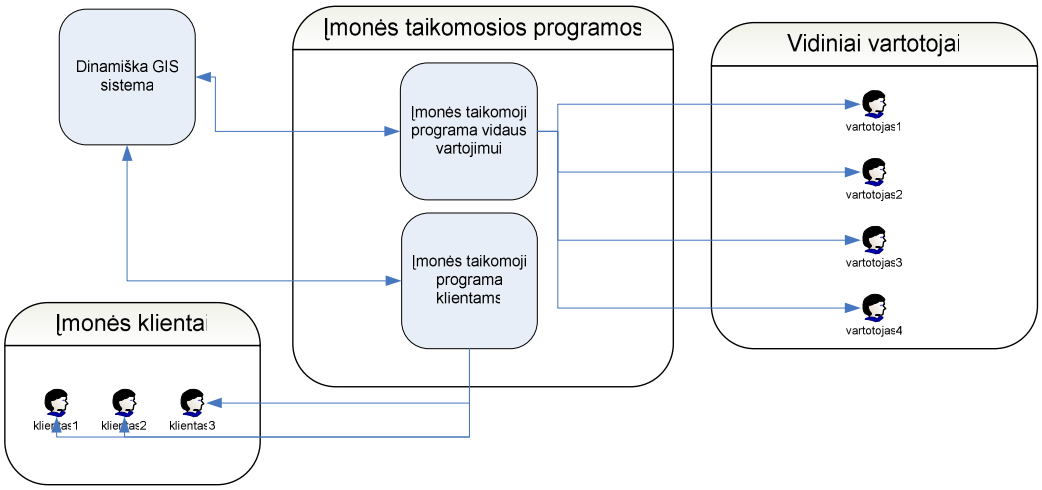
Įmonių teikiamų paslaugų klientams pagerinimas. Daugelis verslo įmonių teikia paslaugas klientams susijusias su žemėlapiais ir už tai gauna pajamas. Paprastai šios įmonės naudojasi kažkieno kito teikiamomis skaitmeninių žemėlapių paslaugomis (pvz. Google

maps), tada atvaizduoja ant žemėlapio tai kas aktualu klientams ir pateikia šio žemėlapio fragmentą. Kiekviena įmonė atvaizduoja skirtingą informaciją skirtingu būdu. Paprastai tai padaryti leidžia žemėlapių paslaugas suteikiančių bendrovių sukurtas API.

Pavyzdžiui, įsivaizduokime įmonę kuri leidžia žemėlapyje stabėti automobilio judėjimą sekdami jį per GPS. Pateikiant žemėlapį, automobilio pozicija parodoma ant žemėlapio nuiešiant apskritimą, toje vietoje kur automobilis yra.

Minėtam pavyzdžiui realizuoti tinka dinamiška GIS sistema, kuri su žemėlapiu susietų reikimą objektą ir pateiktų žemėlapio fragmentą.

Kaip matome iš pateiktų pavyzdžių, sistemos vartojais tampa įmonės, kurios klientams teikia savo paslaugas susijusias su skaitmeniniais žemėlapiais arba įmonės, kurios skaitmeninius žemėlapius naudoja savo vidinėms funkcijoms pagerinti. Tokį vartotoją apibūdina žemiau pateiktas paveikslas 9.



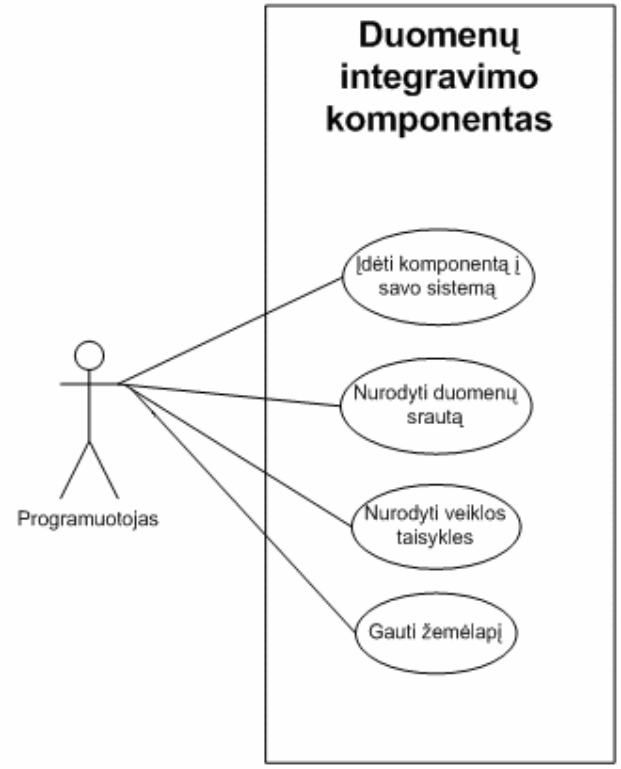
9 pav. Dinamiškos GIS sistemos vartotojai įmonėje

Paveiksle 9 matome, kaip dinamiška GIS sistema naudojasi taikomosios įmonių programos, kuriomis vėliau gali naudotis vidiniai vartotojai arba įmonių klientai. Paveiksle matyti, kad dinamiškos GIS sistemos vartotojas yra įmonės kurios naudojasi šia sistema tam, kad galėtų patenkinti savo vidinius arba klientų poreikius.

2.2.KOMPONENTO PANAUDOS ATVEJAI

Aukščiau pateiktam skyrelyje matėme, kad sistemos teikiamais rezultatais, t.y. žemėlapiais, naudojasi klientai ir firmos vidiniai darbuotojai. Tai yra vartotojai kurie gauna konkretų rezultatą, tačiau jiems nereikia žinoti nieko apie būdą kurio dėka šie rezultatai yra gaunami. Šį rezultatų gavimo būdą realizuoti privalo programuotojas, kadangi jo darbą mes stengiamės palengvinti, tai sukuriame „Duomenų integravimo komponentą“ per kurį

programuotojas galės valdyti dinamišką GIS sistemą. Kadangi siekiame sukurti tokį komponentą, kuris duotų pageidaujamus rezultatus atliekant minimalius programavimo ir konfigūravimo darbus, todėl stengiamės minimizuoti komponento panaudojimo laiko sąnaudas ir sudarome seką elementarių veiksmų, kuriuos programuotojas galės atlikti su komponentu. Veiksmai pateikiami žemiau esančioje panaudos atvejų diagramoje.



10 pav. Duomenų integravimo komponento panaudos atvejų diagrama

Įdėti komponentą į savo sistemą. Duomenų integravimo komponentas yra suprogramuotas kaip laisvai išimama ir įdedama sistemos dalis, todėl programuotojai galės šį komponentą įdėti ir išimti iš savo sistemos neturėdami jokių žinių apie jo veikimo principus. Komponento įdėjimas į sistemą apima kelių kodo eilučių įrašymą internetinio puslapio vietoje, kurioje norėsime, kad atsirastų pageidaujamas žemėlapis su pageidaujamais funkcionalumais. Šios kodo eilutės kreipsis į java servlet‘us kur bus atliekama visa reikalinga logika, kurios pagalba gausime norimą žemėlapį iš nutolusio serverio.

Nurodyti duomenų srautą. Tam, kad galėtume integruoti verslo duomenis su erdvine informacija, reikia juos užkrauti į duomenų integravimo komponentą. Kad realizuotų šį žingsnį, programuotojas gali nurodyti duomenų srautą, kuriame bus duomenys kuriuos reiks apdoroti ir pagal rezultatus sugeneruoti žemėlapį. Srautas nurodomas tiesioginio prisijungimo

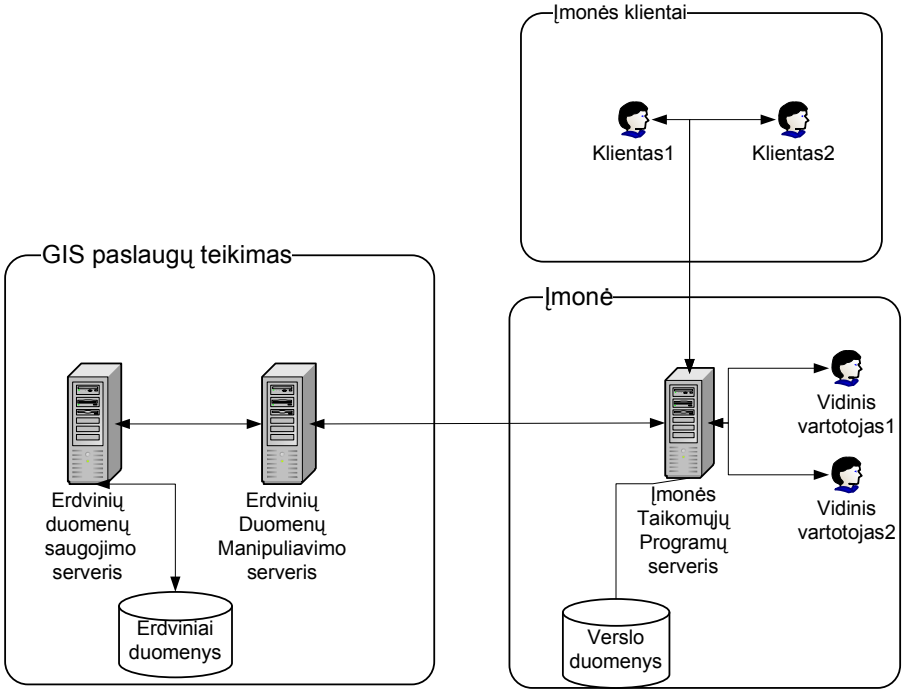
prie duomenų bazės per JDBC tvarkyklę būdu. Taigi akivaizdu, kad programuotojui nereikia rūpintis duomenų užkrovimu į atmintį. Jam tereikia nurodyti parametrus, per kuriuos komponentas pats prisijungs prie verslo duomenų bazės. Tokiu būdu tereikia tik užtikrinti, kad duomenys verslo duomenų bazėje bus paruošti taip, kaip nurodyta skyrelyje „Duomenų struktūra“.

Nurodyti veiklos taisykles. Vien tik užkrauti verslo duomenis nepakanka. Toliau programuotojas gali nurodyti eilę veiklos taisyklių, kurias interpretuodamas algoritmas filtruos verslo duomenis išrinkdamas tuos kuriuos reikia sugeneruoti žemėlapyje. Kitais žodžiais tariant programuotojui tereikia „aprašyti savais žodžiais“ tai ką jis nori matyti. Veiklos taisyklės, kaip ir duomenų srauto nurodymas, komponentui paduodami per parametrus laisvai pasirenkant programuotojo.

Gauti žemėlapį. Tinkamai nurodžius veiklos taisykles ir duomenų srautą, įkraunant internetinį puslapį kuriame įdėtas komponentas, įkraunamas ir žemėlapis pagal nurodytus parametrus. Programuotojas nedalyvauja žemėlapio generavimo procese, nes tai yra galutinis rezultatas priklausantis nuo minėtų parametrų nustatymo.

2.3.SISTEMOS VEIKIMO PRINCIPAS

Kuriamoje sistemoje stengiamės maksimaliai atskirti vieną nuo kito atskirus komponentus. Natūralu, kad erdvinės informacijos saugykla turi būti tik viena, nes tokia informacija yra brangi ir užima ištisus terabaitus serverių atminties. Tuo tikslu erdvinė informacija ir programinė įranga skirta jos apdorojimui laikoma nutolusiuose serveriuose. Tuomet daugelis įmonių iš skirtingų pasaulio vietų gali naudotis šia viename taške sukoncentruota informacija. Žemiau pateiktame paveiksle 11 matome sistemos veikimo principo paveikslą kuriame parodyta kaip atskiros dalys siejasi tarpusavyje tam, kad „klientas“ ir „vidinis vartotojas“ gautų informaciją kurios jam reikia:



11 pav. Sistemos veikimo principas

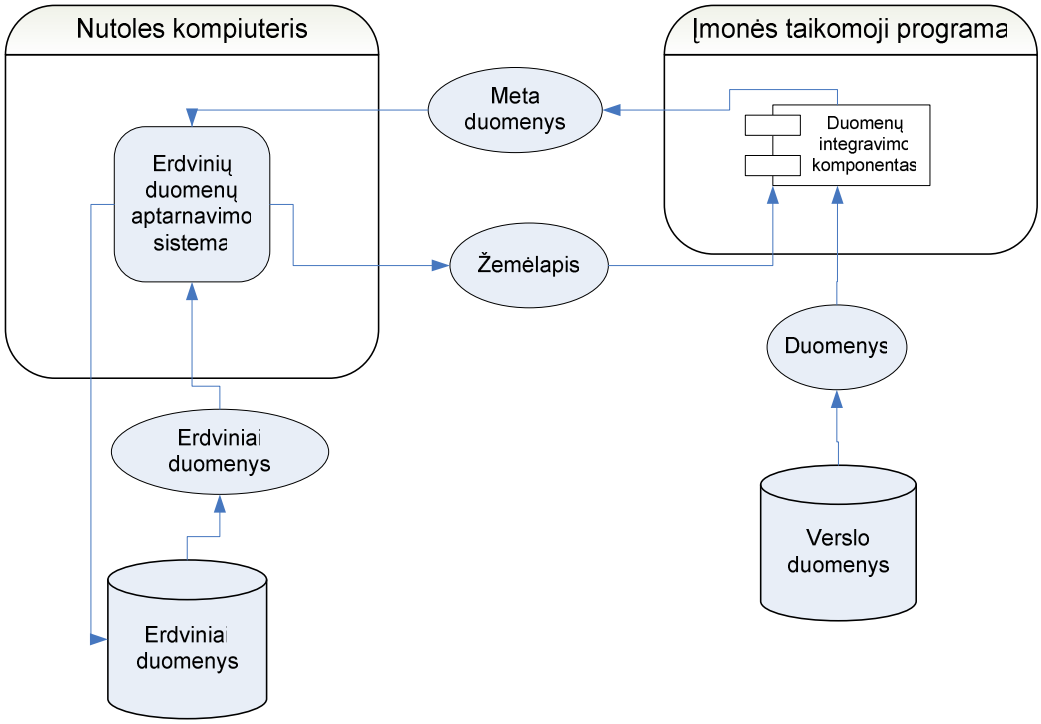
Veikimo principo aiškinimą pradėkim iš kliento pusės: klientas atsidaro interneto naršyklę ir suveda kažkokios įmonės teikiamos paslaugos (kuri naudoja žemėlapius) adresą. Tuomet įmonės taikomųjų paslaugų serveris gauna užklausą iš kliento ir suaktyvina toje taikomojoje paslaugoje įprogramuotą komponentą. Tuomet komponentas pasinaudoja jam nurodytais duomenų srauto parametrais ir užkrovęs verslo duomenis juos apdoroja pagal veiklos taisykles. Iš šių duomenų išskyręs aprašomuosius (meta) duomenis (skyrelis „metaduomenų struktūra“), komponentas juos siunčia į erdvinių duomenų manipuliavimo serverį, šis kreipiasi į serverį kuriame saugoma visa erdvinė informacija ir grąžina reikalingus erdvinius duomenis. Iš šių erdvinių duomenų sugeneruojamas žemėlapis ir grąžinamas į įmonės taikomųjų paslaugų serverį. Čia sugeneruojamas internetinis puslapis su gautu žemėlapiu kuris užkraunamas į kliento naršyklę.

2.4.ERDVINIŲ IR VERSLO DUOMENŲ SRAUTAI

Dinamiškai generuojant žemėlapį, neišvengimai įvyksta tam tikrų dalių verslo ir erdvinių duomenų integracija. Kadangi šie duomenys yra laikomi atskirai nutolusiuose kompiuteriuose, todėl jie neturi būti suintegruojami fiziškai. Naudodamas sąvoką „integravimas“ turiu galvoje, kad verslo duomenys nėra niekur siunčiami, jie apdorojami įmonės taikomosios programos pusėje. Akivaizdu, kad visiškai nėra poreikio visų verslo duomenų siuntimui į nutolusį kompiuterį dėl dviejų priežasčių:

- Nutolusiame kompiuteryje saugomi tik erdviniai duomenys. Todėl praplėsti juos ir išsaugoti ateičiai kažkuriam verslui būdinga specifine neerdvine informacija nėra tokios sistemos tikslas. Nutolusiame kompiuteryje saugoma tiktai erdvinė informacija kurios neturi įmonės ir ši pusė nieko nežino apie tai su kokiais verslo duomenimis ji bandoma integruoti.
- Gali užtrukti labai daug laiko persiųsti didelius kiekius verslo duomenų į nutolusį serverį.

Lygiai taip pat įmonės taikomoji programa nežino nieko apie tai kokie erdviniai duomenys ir kiek jų yra nutolusiame serveryje. Integruojant duomenis yra tikimasi, kad verslui reikalingi duomenys yra nutolusiame erdvinės informacijos saugojimo serveryje. Duomenų integracija žemėlapio generavimui pateikta žemiau esančiame paveiksle 12.



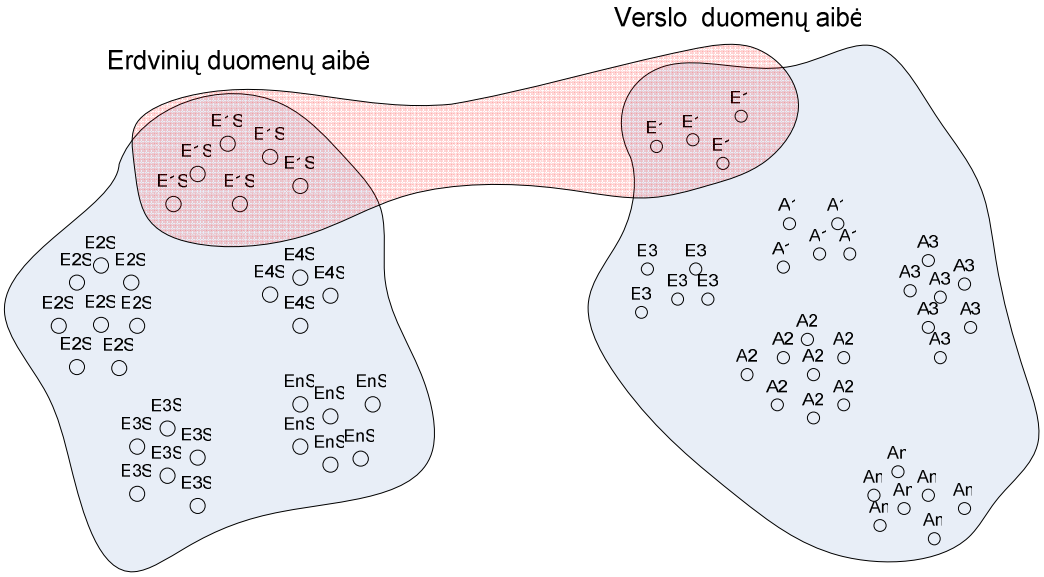
12 pav. Duomenų srautai tarp sistemos komponentų

12 paveiksle pateikiama, kaip vyksta duomenų integravimo ciklas ir kokie yra duomenų srautai. Paveikslą nagrinėti pradėkime nuo verslo duomenų bazės. Kai yra pareikalaujama išanalizavus verslo duomenis sugeneruoti žemėlapį atsiranda tokie duomenų srautai: Į duomenų integravimo komponentą užkraunami verslo duomenys. Šis duomenų kiekis yra sąlyginai didelis, todėl duomenų integravimo komponentas juos apdoroja ir išskiria metaduomenis, kurie aprašo kaip turi būti sugeneruojamas žemėlapis. Metaduomenų kiekis yra sąlyginai mažas, todėl jie gali būti nusiųsti į erdvinių duomenų aptarnavimo sistemą

esančią nutolusiame kompiuteryje. Galima sakyti, kad nutolusiame kompiuteryje ir įvyksta vadinamasis virtualus duomenų integravimas, kadangi atsižvelgiant į gautus verslo metaduomenis, užkraunami atitinkami erdviniai duomenys ir sugeneruojamas žemėlapio fragmentas. Sugeneruotas žemėlapis persiunčiamas atgal į įmonės taikomosios programos serverį. Tokiu būdu pasinaudojant verslo ir erdviniais duomenimis buvo gautas žemėlapis kuris apima tiek erdvinių tiek verslo duomenų vaizdavimą, todėl galima sakyti, kad buvo atlikta virtuali duomenų integracija, tačiau integravimo rezultatas niekur neįsimintas, o tik sugeneruotas žemėlapis.

2.5.REIKALAVIMAI ERDVINIAMS IR VERSLO DUOMENIMS

Ankstesniuose skyriuose buvo minėta, kad dinamiška GIS sistema suteikia galimybę aprašomuosius duomenis integruoti su erdviniais duomenimis. T.y. skirtingos įmonės įsidiegs šią sistemą ir norės matyti ekrane tik pagal jų verslui būdingus duomenis sugeneruotą žemėlapį, nieko nežinodami apie tai kaip ir kokie erdviniai duomenys bus parinkti pagal jų verslo duomenis. Tam, kad įvyktų sėkminga integracija ir būtų sugeneruotas žemėlapis, verslo ir erdviniai duomenys turi tenkinti sąlygas nurodytas žemiau esančiame paveiksle 13.



13 pav. Erdvinių ir verslo duomenų aibės

Paveiksle matome dvi duomenų aibes: erdvinių ir verslo. Verslo duomenų aibėje matome saugomus duomenų tipus E1, E3 (pavyzdžiui E1 atitinka kelius, o E3 - ežerus). Tokiu atveju E1 ir E3 laukai yra nekas kita kaip kelių ir ežerų pavadinimai apie kuriuos verslas surinkęs aprašomąją informaciją A1, A2, A3... An. Tokia aprašomoji informacija gali būti keliuose įvykusių avarijų skaičius, vidutinė ežero temperatūra, kelio dangos tipas ir t.t. Erdvinių duomenų aibėje turime tuos pačius duomenų tipus E1S, E2S... EnS. Čia

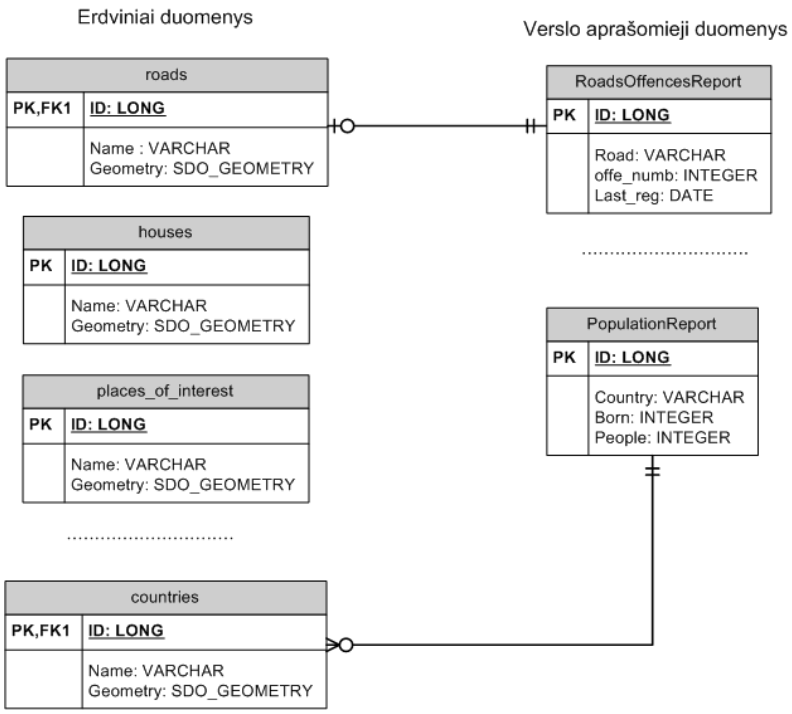
pažymėjimas E1S reiškia, kad tai yra E1 tipo duomenys kartu su erdvine (S) informacija. Taigi šiuo atveju E1S atitiktų saugomus absoliučiai visus mums žinomus kelius bei jų koordinates kurios atitinka realias pozicijas žemės paviršiuje.

Apvesta aibių sankirta reiškia, kad jei norėsime matyti žemėlapyje informaciją susijusią su verslo duomenimis E1, tai turime būtinai turėti šių duomenų erdvinį aprašą erdvinių duomenų aibėje, kad galėtume gauti koordinates žemėlapio generavimui. Kitaip tariant verslo duomenų aibė atmetus aprašomuosius duomenis yra erdvinių duomenų poaibis.

Jeigu erdvinių duomenų aibėje neturime reikiamų duomenų, integracija neįvyksta ir gaunamas žemėlapis be duomenų kurių neradome. Tačiau vėliau trūkstamais duomenimis papildžius erdvinių duomenų aibę programos kodo keisti nereikia ir automatiškai integravimo procesas taps sėkmingas kadangi aibė bus papildyta trūkstamais elementais.

2.6. REIKALAVIMAI ERDVINIŲ IR VERSLO DUOMENŲ STRUKTŪRAI

Ankstesniame skyrelyje aptarėme į kokias aibes yra suskirstyti duomenys dinamiškoje GIS sistemoje. Toliau detalizuosime, kokios struktūros turi būti duomenys erdvinių ir verslo duomenų aibėse. Kaip jau buvo minėta erdvinių duomenų aibė nežino kaip ir kokie duomenys yra saugomi aprašomųjų duomenų aibėje ir atvirkščiai, aprašomųjų duomenų aibė nežino kokie duomenys yra erdvinių duomenų aibėje. Duomenų saugojimo struktūra tampa aktuali tuomet, kai kalbame apie duomenų integravimo komponentą ir žemėlapių generavimą. Žemėlapių generatoriui turi būti žinoma erdvinių duomenų struktūra, o duomenų integravimo komponentui turi būti žinoma verslo duomenų struktūra. Taigi šios struktūros yra pateikiamos žemiau esančiame paveiksle 14:



14 pav. Duomenų struktūra

Paveikslo kairėje pusėje matome erdvinių duomenų lentelių struktūrą, o dešinėje matome verslo duomenų struktūrą. Pavyzdiniame paveiksle pateiktas tik formatas kaip duomenys turi būti saugomi. Realiu atveju lentelių skaičius tiek erdvinių duomenų tiek ir verslo duomenų pusėje nėra baigtinis, kadangi galime turėti daug aprašomųjų arba erdvinių duomenų saugomų skirtingose lentelėse ar duomenų bazėse, tačiau pagal sutartą struktūrą. Paveiksle pavaizduotos rodyklės nereiškia, kad lentelės yra sujungtos realiu ryšiu, rodyklės parodo tik tai, kad atitinkamose erdvinių duomenų lentelėse yra saugomi duomenys kuriuos aprašo atitinkamos verslo aprašomųjų duomenų lentelės. Realiai šios lentelės saugomos skirtingose nutolusiose duomenų bazėse, todėl ryšys tarp jų yra negalimas.

Erdvinių duomenų struktūra. Erdviniai duomenys yra nematomi nei programuotojui, nei pačiam duomenų integravimo komponentui. Šiuos duomenis apdoros Oracle spatial variklis, todėl jie turi išlaikyti griežtai tokią struktūrą kuri nurodyta 14 paveiksle. Matome, kad erdvinius duomenis gali sudaryti *n* lentelių kuriose saugomi skirtingo tipo erdviniai duomenys: jų pavadinimai ir jų erdviniai geometriniai parametrai. Oracle spatial tam skiria objektą *SDO_GEOMETRY*, kuriame saugomos iš taškų sudarytos figūros, atitinkančios realias koordinates. Pavyzdžiui, vienoje lentelėje galime saugoti tik kelius, kitoje tik miškus, dar sekančioje – ežerus ir t.t. Taip galima turėti visą eilę erdvinių sluoksnių kurios galėsime naudoti žemėlapių generavimui. Kurioje lentelėje koks duomenų tipas saugomas, aprašys metaduomenys pateikiami „metaduomenų struktūra“ skyrelyje. Pavyzdiniu atveju matome, kad turime erdvinę informaciją apie kelius, namus, šalis ir lankytinas vietas. Šis duomenų

kiekis nėra baigtinis ir realiai galime turėti dešimtis arba šimtus lentelių kurios aprašo skirtingus erdvinius duomenis.

Verslo duomenų struktūra. Verslo duomenys kuriuos reikia užkrauti į duomenų integravimo komponentą, gali būti saugomi skirtingose duomenų bazėse ir skirtingose lentelėse. Šiuos duomenis sudaro: erdvinis objektas ir kiek norima laukų aprašomosios informacijos. Erdvinis objektas nurodo erdvinės informacijos tipą apie kurią turim sukaupę aprašomosios informacijos. Pavyzdžiui, pavyzdiniame paveiksle turim lentelę kurioje saugoma nusižengimų keliuose ataskaita („raodsOffencesReport“). Šioje ataskaitoje duomenų tipas yra laukas „road“ kuris žymi kelią apie kurį turim sukaupę aprašomosios informacijos. Čia yra paprasčiausiai saugomas kelio pavadinimas *VARCHAR* tipo laukelyje. Pagal šį pavadinimą bus ieškoma kelio atitikmens erdviniuose duomenyse. Toliau nurodomi aprašomieji laukai *offe_numb* ir *last_reg*. Jie atitinkamai aprašo pažeidimų kelyje kiekį ir kada buvo užregistruotas paskutinis pažeidimas. Šie laukai gali būti kokio norima tipo (Integer, DATE ir t.t.). Pagal šiuos laukus duomenų integravimo komponentas nuspręs ar reikia vaizduoti žemėlapyje konkretų kelią ar ne. Pavyzdiniu atveju turim dvi aprašomųjų duomenų lenteles, tačiau realiu atveju jų skaičius yra neribojamas ir klientas gali nurodyti prisijungimą prie dešimčių ar šimtų lentelių esančių skirtingose duomenų bazėse. Kaip yra konfigūruojami prisijungimai prie aprašomųjų įmonės duomenų bazių paaiškinta 2.7 skyrelyje.

2.7. METADUOMENŲ STRUKTŪRA

Sistemos metaduomenis sudaro erdvinių duomenų elementų tipai. Natūralu, kad erdvinių duomenų aibėje konkretaus tipo elementų grupė (pvz.: keliai) saugomi kažkokiu pavadinimu pavadintoje lentelėje ir tuo pat metu verslo duomenų pusėje, tie patys elementai yra vadinami kitaip. Todėl parengiant verslo duomenis integravimui reikia nurodyti duomenų tipą kuris apibrėžiamas kaip meta duomenys. Tai yra paprasčiausi sutartiniai pažymėjimai kaip pavyzdžiui, kelius atitiktų kodas – *spat_roads*, miškus – *spat_forests*, ežerus – *spat_lakes* ir t.t. Šie kodai iš anksto bus parengti komponento dokumentacijoje ir neišvengiamai privers vartotoją tinkamai aprašyti savo verslo duomenis. Tokiu būdu duomenų tipas erdvinių duomenų aibėje galės būti vadinamas vienokiu vardu, o verslo duomenų aibėje kitokiu vardu, tačiau bet kokiu atveju šie abu pavadinimai turės bendrą kodą per kurį galėsime juos susieti. Tokiu būdu yra tinkamai išsprendžiama skirtingų šalių kalbos problemos ir pan.

Atliekant verslo duomenų filtravimą į žemėlapių generavimo serverį keliauja atfiltruota informacija, kurią sudaro elementą apibūdinantys meta duomenys, bei maksimaliai supaprastinta aprašomoji informacija, kurią yra būtina atvaizduoti ant žemėlapio. Taip pat yra sugeneruojami elementų atvaizdavimo būdo (spalvos šešėliai ir t.t.) žemėlapyje metaduomenys. Šie duomenys priklauso nuo naudojamo žemėlapių generavimo variklio (mūsų atveju MapViewer) ir jie yra žinomi tik duomenų integravimo komponentui. Žemėlapių generavimo varikliui visi minėti metaduomenys ir aprašomoji informacija pateikiama dinamiškai sugeneruotų komandų seka. Toks komandų generavimas yra paremtas veiklos taisyklių koncepcija.

Norimų objektų atvaizdavimo būdui (t.y. nurodant atvaizdavimo spalvą) yra iš anksto parengtos galimos spalvų grupės kurias integravimo komponentas veiklos taisyklėse atpažįsta kaip spalvas ir objektus ant žemėlapio nuspalvina šia spalva.

Atvaizdavimo kriterijui nurodyti (t.y. matematinė taisyklė pagal kurią lyginami aprašomieji duomenys) yra iš anksto parengti matematiniai šablonai, kurie atpažįstami veiklos taisyklėse. Šiuos šablonus galima derinti tarpusavyje skirtingose veiklos taisyklėse nurodant skirtingas matematines palyginimo išraiškas pagal kurias turi būti atrinkti įmonės aprašomieji duomenys.

Matematiniuose šablonuose galima naudoti iš anksto nustatytus objektus pagal kuriuos yra atpažįstama ar palyginamas kriterijus yra data, laikas, skaičius, tekstas ir t.t.

Šiame skyrelyje aptarti galimi atvaizdavimo būdai, matematiniai šablonai ir objektai kuriuos galime naudoti veiklos taisyklėse yra pateikiami 2 priede.

Norint, kad komponentas žinotų iš kur reikia pasiimti įmonės aprašomuosius duomenis, reikia aprašyti metaduomenis, kurie nusako kur kokių duomenų reikia ieškoti. Reikia nurodyti tokius parametrus.

- Prisijungimo prie duomenų bazės adresą – adresas kuriame serveryje duomenų bazė yra.
- Duomenų bazės tipą – kokia tai duomenų bazė (galimi variantai: MySQL, ORACLE, MsSQL, PostresQL)
- Duomenų bazės vardą – vardas duomenų bazės kuriame saugoma aprašomoji informacija
- Lentelės vardą – lentelė kurioje saugoma aprašomoji informacija
- Erdvinio objekto lauką – lentelės laukelio pavadinimas kuriame saugomas erdvinio objekto pavadinimas (pvz. kelio pavadinimas)
- Erdvinio objekto vardą – vardas kuriuo naudodamiesi į šį objektą kreipsimės iš veiklos taisyklių

- Erdvino objekto metatipą – nurodoma su kokiais erdviniais duomenimis reiktų integruoti erdvini objektą norint gauti jo poziciją erdvėje (pvz. jei įrašysim spat_roads tai komponentui nurodysim, kad jis ieškotų kelių)
- Aprašomojo parametro lauką – lentelės laukelio pavadinimas kuriame saugomas aprašomojo atributo vardas (pvz. pažeidimų kelyje skaičius)
- Aprašomojo parametro vardą – vardas kuriuo naudodamiesi kreipsimės į aprašomąjį parametrą ir veiklos taisyklės matematinės išraiškos.

Konfigūruojant metaduomenis galima turėti neribotą skaičių šių aukščiau išvardintų parametrų porų, todėl galima aprašyti neribotą kiekį duomenų bazių ir jose saugomų aprašomųjų duomenų pagal kuriuos norime sugeneruoti žemėlapį.

2.8.NEFUNKCINIAI REIKALAVIMAI SISTEMAI

Nefunkciniai reikalavimai yra svarbi sistemos dalis ir jie tiesiogiai įtakoja sistemos patrauklumą patogumą ir panaudojimo efektyvumą. Šioje vietoje išskiriame du pagrindinius nefunkcinius reikalavimus: *našumas* ir *panaudojimo efektyvumas*.

Našumas. Kadangi daugybė klientų naudosis vienu erdvinių duomenų informacijos šaltiniu, todėl turi būti užtikrintas greitas erdvinių duomenų apdorojimas ir žemėlapių generavimas. Daugeliu atvejų vartotojų užklausos gali reikalauti didelio informacijos kiekio peržiūros erdvinėje duomenų bazėje, todėl reikia tinkamai sukonfigūruoti šią informaciją aptarnaujančią programinę įrangą. Erdviniams duomenims reikia taikyti erdvinius indeksus ir atitinkamai grupuoti į atskiras grupes, taigi yra svarbu projektuoti sistemą taip, kad erdvinių duomenų analizė vyktų kuo sparčiau, ir kuo mažiau apkrautų erdvinės informacijos apdorojimo serverius.

Generuojant erdvinius žemėlapius vienu metu gali tekti apdoroti daugybe vartotoju užklausų, todėl vieną kartą sugeneruoti žemėlapiai turi būti išsaugoti tam tikram laikotarpiui, kad nerikėtų dar kartą generuoti tokio pat žemėlapio ir apkrauti GIS.

Svarbu yra užtikrinti kad kiekvieno kliento užklausos būtų vykdomos lygiagrečiai nepriklausomai viena nuo kitos, tokiu būdu jei vieno kliento užklausa trunka ilgai arba patenka į mirties tašką – nenukentėtų kitų klientų užklausoms apdoroti skirtas laikas.

Visi šie reikalavimai turi būti pasiekti tinkamai sukonfigūravus Oracle DBVS programinę įrangą kuri pakankamai funkcijų minėtoms problemoms spręsti.

Panaudojimo efektyvumas. Lengviausia ir patogiausia būtų naudotis tokia sistema, kuri reikalauja minimalių vartotojo žinių ir pastangų norint ją pasinaudoti. Duomenų integravimo komponentas turi pasižymėti tokiomis savybėmis. Tai pasiekiama tokiais būdais:

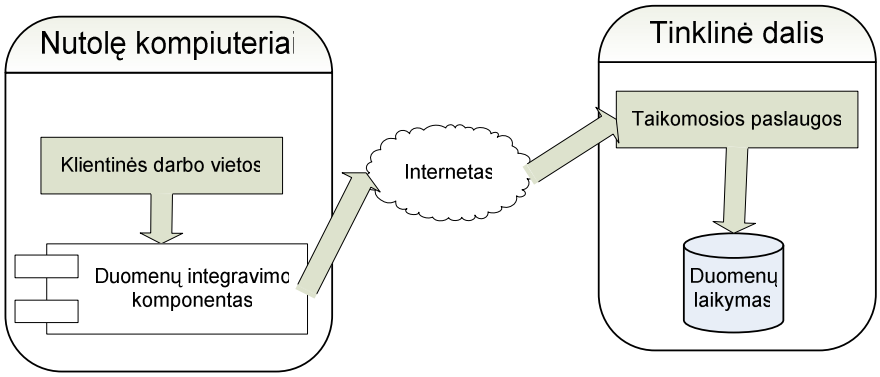
- Programavimo darbų minimizavimas. Norėdamas įdiegti žemėlapius į savo internetinį puslapį programuotojas turės parašyti labai minimalų kodo kiekį.
- Konfigūravimo patogumas. Programuotojai konfigūruos komponentą naudodami veiklos taisykles, t.y. natūralia kalba nusakys tai ką nori matyti ant žemėlapio.
- Duomenų pateikimo patogumas. Programuotojui nereikės atitinkamai parengti duomenų atmintyje, pakaks nurodyti tiesioginį prisijungimą prie duomenų bazės.

3. DINAMIŠKOS GIS SISTEMOS ARCHITEKTŪRA

3.1. BENDRA SISTEMOS ARCHITEKTŪRA

Dinamiškos GIS architektūra yra sudaryta iš keturių atskirų komponentų. Kiekvienas komponentas atlieka tik jam pavestą funkciją bendraudamas su kitu komponentu ir rezultate gauname dinamiškai suintegruotus verslo ir erdvinius duomenis iš kurių sugeneruojamas žemėlapis. Žemiau pateikiami atskirų komponentų aprašymai:

- Duomenų laikymas – apima įvairių erdvinių ir aprašomųjų duomenų sandėliavimą organizacijos viduje. Šitie duomenų sandėliai paprastai turi savyje informaciją keliuose nesuderinamuose duomenų formatuose.
- Taikomosios paslaugos - susideda iš GIS Taikomojo Serverio ir Tinklo Serverio, kurie yra atsakingi už duomenų prieigą, apdorojimą ir vartotojo aptarnavimą per kliento darbo vietą (klientų programinę įrangą). Tinklo Serveris, tai GIS Tinklo Serveris, kuris leidžia komunikuoti su vartotojų standartiniais interneto serveriais tokiais kaip Apache, Microsoft IIS ir pan. Taikomasis GIS Serveris yra funkcinis GIS paslaugų pagrindas jis suteikia erdvinių duomenų integravimą ir transformavimą.
- Klientinė darbo vieta – klientais gali būti visi informacinių sistemų vartotojai esantys bet kuriame pasaulio taške. Klientas paprastai kreipiasi į GIS taikomąsias paslaugas prašydamas atvaizduoti tam tikrą erdvinių duomenų sluoksnį ir jame esančius objektus. Kliento darbo vieta gali buti interneto naršyklės, mobilieji įrenginiai ir t.t.
- Duomenų integravimo komponentas - tai unikalus komponentas, kuris leidžia dinamiškai generuoti žemėlapius iš erdvinių ir verslo duomenų. Tokio komponento egzistavimas daro GIS sistemą unikalia ir dinamiška lyginant su kitais egzistuojančiais GIS sprendimais.

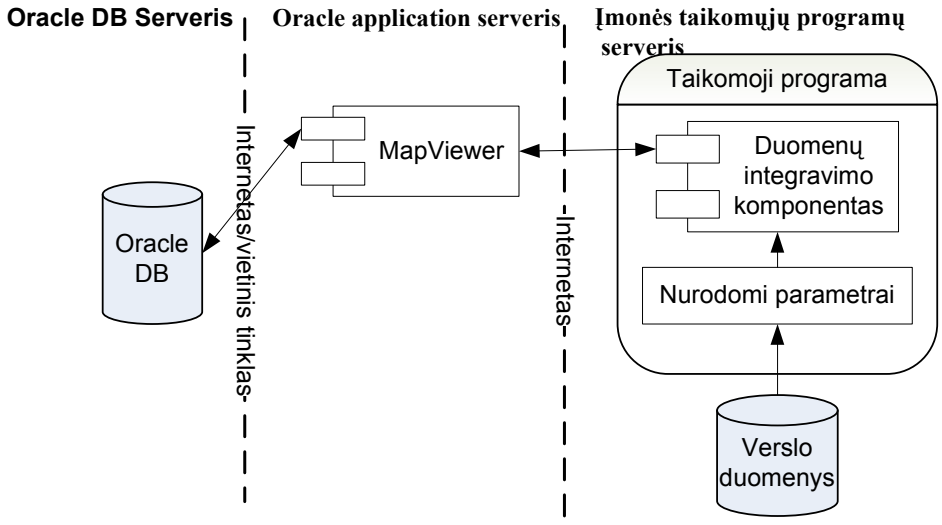


16 pav. Bendra sistemos architektūra

Pateiktam paveiksle matome, kad tokia architektūra leidžia turėti vieną erdvinių duomenų šaltinį ir daug klientinių taikomųjų programų kuriose yra įdiegtas duomenų integravimo komponentas. Tai yra labai patogu, kadangi erdvinė informacija yra brangi ir nevisiems asmeniškai įperkama, todėl iš vieno taško bus galima aptarnauti daugelį klientų užklausų. Tinklinė dalis nebus apkrauta verslo duomenų analizės darbais, o tai, be abejo užtikrina greitą atsakymo į kliento užklausą laiką.

3.2. ARCHITEKTŪRA PANAUDOJANT ORACLE DBVS

Kadangi darbe tiriame Oracle programinės įrangos GIS kūrimo galimybes, todėl minėtos architektūros realizacijai parinksime atitinkamą Oracle programinę įrangą. Jau eilę metų Oracle yra absoliutus lyderis DBVS rinkoje, todėl atskiri produktai suteikia maksimalų efektyvumą ir patogumą. Ši programinė įranga yra įdiegiama nutolusiuose kompiuteriuose kur saugoma ir apdorojama erdvinė informacija. Tokiu būdu vartotojams, kurie naudosis dinamiška GIS sistema, nereikės savo pusėje įsidiegti jokios programinės įrangos, tereikės savo programoje įdėti duomenų integravimo komponentą, kuris pats bendraus su nutolusiu kompiuteriu interneto pagalba. Toks sprendimas panaudojant Oracle DBVS pateikiamas žemiau esančiame paveiksle 17.



17 pav. Sistemos architektūra panaudojant Oracle DBVS

Oracle DB serveris – Oracle DBVS serveris, kuriame saugomos duomenų bazės su erdvine informacija.

Oracle application serveris – taikomųjų uždavinių serveris, kuriame įdiegtas MapViewer komponentas, turintis galimybę komunikuoti su **Oracle DataBase** serveriu ir naudoti erdvinius duomenis žemėlapių generavimui.

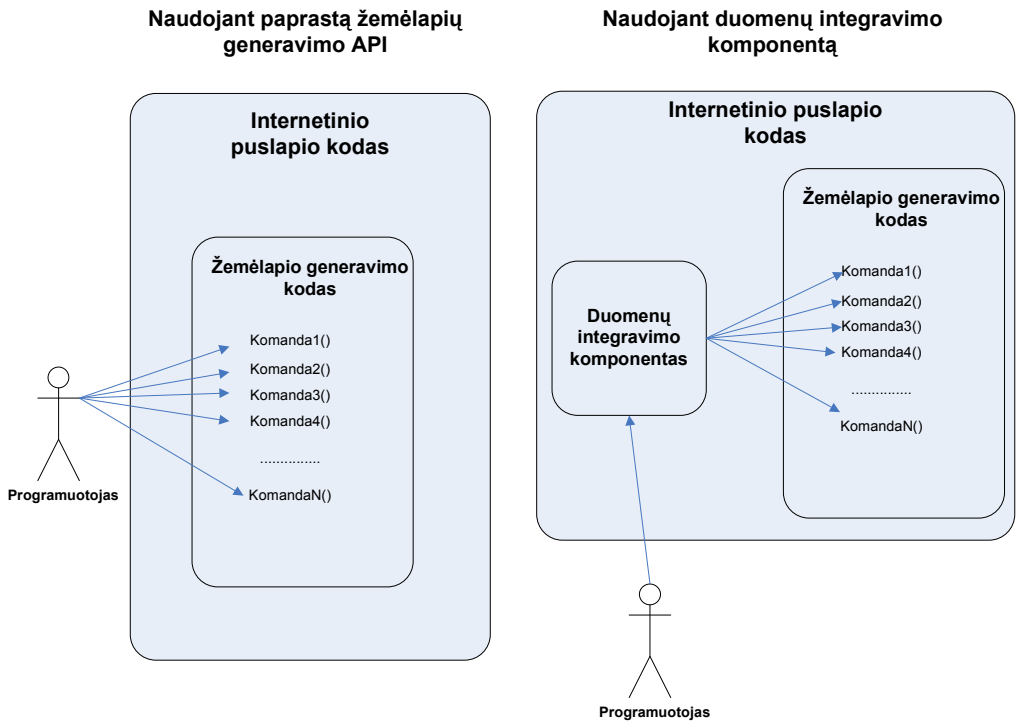
Įmonės taikomųjų programų serveris – šiame serveryje yra vartotojo taikomojo programa, kuri naudoja duomenų integravimo komponentą. Vartotojo taikomoji programa aprūpina komponentą reikalingais parametrais, kad šis galėtų atlikti duomenų integravimo funkciją.

3.3.ARCHITEKTŪROS NAUJUMAS LYGINANT SU KITOMIS GIS

Kaip minėta ankstesniame skyrelyje, architektūros naujumas apima duomenų integravimo komponentą, kuris grindžiamas veiklos taisyklių koncepcija. Šis komponentas turi atlikti tokias funkcijas:

- Užkrauti verslo duomenis.
- Interpretuoti nurodytas veiklos taisykles
- Atlikti duomenų filtraciją pagal veiklos taisykles.
- Sugeneruoti komandų sekas pagal kurias būtų generuojamas žemėlapio fragmentas nutolusiame serveryje.

Detaliau panagrinėsime komandų sekos generavimo principą tam, kad parodytume šio komponento žemėlapių generavimo dinamiškumą. Daugelis šiuo metu egzistuojančių žemėlapių generavimo sistemų suteikia API kurio pagalba galima gauti norimą žemėlapį. Oracle MapViewer yra ne išimtinis atvejis. MapViewer suteikia keletą žemėlapių generavimo API. Pavyzdžiui: JSP atributų rinkinį, XML API ir JAVA script API. Žemėlapių valdymui pasirenkame JAVA script API, kadangi mūsų tikslas yra sukurti komponentą kuris gali būti laisvai įdedamas į internetinius puslapius. Kaip nepaslaptins tokius API suteikia ir daugiau žemėlapių tiekėjų, tarp jų ir rinkos lyderis Google maps. Visi šie API turi vienodą panaudojimo struktūrą: norėdami pasinaudoti žemėlapiais programuotojai perskaito šių API dokumentaciją ir suprogramuoja kodą, kuris generuoja žemėlapius. Pasikeitus užsakovo reikalavimams jie kodą perrašo pagal naujus reikalavimus. Žemiau esančiame paveiksle 18 pateikiamas minėtos situacijos palyginimas naudojant paprastus API ir duomenų integravimo komponentą.



18 pav. Architektūros naujumas

Pateiktame paveiksle 18 matome du žemėlapių įdėjimo į internetinį puslapį būdus. Matome, kad pirmuoju būdu programuotojas pats turi parašyti kiekvieną žemėlapio generavimo komandą ir kiekvienoje komandoje pats nurodo kokius ir kaip verslo duomenis pageidauja atvaizduoti, atlikdamas jų filtraciją ir apdorojimą. Antrame paveiksle matome kaip tas pats žemėlapis gali būti sugeneruotas nauju, šiame darbe siūlomu būdu. Matome, kad programuotojas įdeda duomenų integravimo komponentą į interneto puslapį, bei tinkamai jį sukonfigūruoja. Tada komponentas dinamiškai sugeneruoja komandų seką, kuri realizuoja žemėlapio generavimo darbus. Matome, kad antruoju atveju programuotojas visiškai išlaisvinamas nuo programavimo darbų ir jam pačiam nieko nereikia žinoti apie komandas, kurių pagalba generuojamas žemėlapis, kadangi tai atlieka duomenų integravimo komponentas.

3.4.KOMPONENTO REALIZACIJOS PRIEMONIŲ PASIRINKIMAS

Komponento realizacijai naudosime java Web technologiją. Verslo duomenų užkrovimui iš duomenų bazės naudosime JDBC tvarkykles. Numatyti integruoti prisijungimą prie PostgreSQL, Oracle, MySQL ir MS SQL duomenų bazių panaudojant atitinkamas prisijungimą prie bazių leidžiančias java (JDBC) bibliotekas. Komponentą bus galima įdėti į vartotojų java kodą. Komponento sugeneruotą žemėlapį grąžins java taikomoji programa

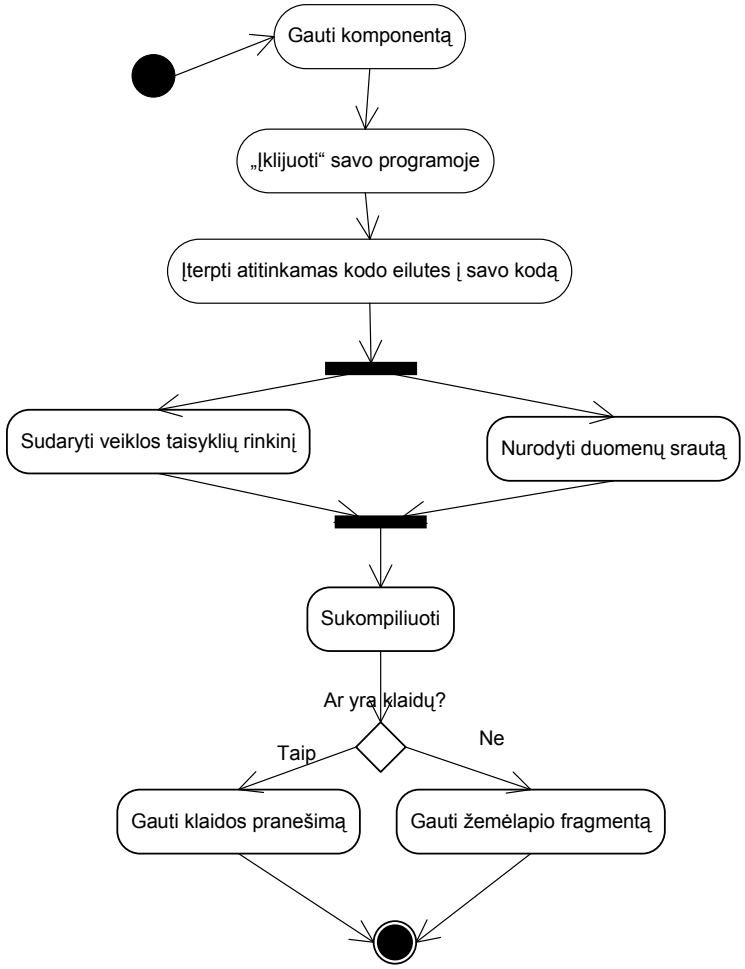
kurioje bus realizuotas veiklos taisyklių interpretavimo variklis. Atskiros funkcijos bus suprogramuotos atskirose *.java bylose. Komponento testavimui sukursime internetinį puslapį panaudojant JSP, JSF, IceFaces internetinių puslapių kūrimo technologiją.

4. DUOMENŲ INTEGRAVIMO KOMPONENTO PROJEKTAS

Šis komponentas atlieka verslo duomenų parengimą integravimui su erdviniais duomenimis. Kadangi daugelis įmonių žemėlapius nori vaizduoti internetiniuose puslapiuose, bei suteikti tam tikrą žemėlapių ir objektų juose valdymą, komponentas suteikia sąsają, kurios dėka įmonės internetiniame puslapyje būtų galima pavaizduoti žemėlapį bei suteikti jam norimą funkcionalumą. Komponentas turi metodus, kurie susieja verslo duomenis su erdviniais duomenimis nereikalaujant didelių vartotojo pastangų. T.y. vartotojas nežino ir neturi nieko bendra su tuo, kokiu būdu ir kokiam pavidale yra saugomi erdviniai duomenys. Jis žino tik tai tą, ką jis nori matyti. Šiam tikslui pasiekti, galima iš anksto parengti veiklos taisyklių šablonus, kurių dėka verslo duomenys būtų susiejami su erdvine informacija. Veiklos taisyklių aprašymui reiktų naudoti standartizuotą metodą. Vienas iš tokių priimtiniausių metodų būtų Business Rules Solutions RuleSpeak [16]. Šis metodas turi nestruktūrinių veiklos taisyklių aprašymo šablonais galimybę. Erdvinių ir verslo aprašomųjų duomenų integravimui nestruktūrizuotų veiklos taisyklių pilnai pakanka.

4.1. KOMPONENTO DIEGIMAS

Kadangi šis komponentas yra skirtas programuotojams detaliau pateikiamas šio komponento panaudojimas iš programuotojo pusės:

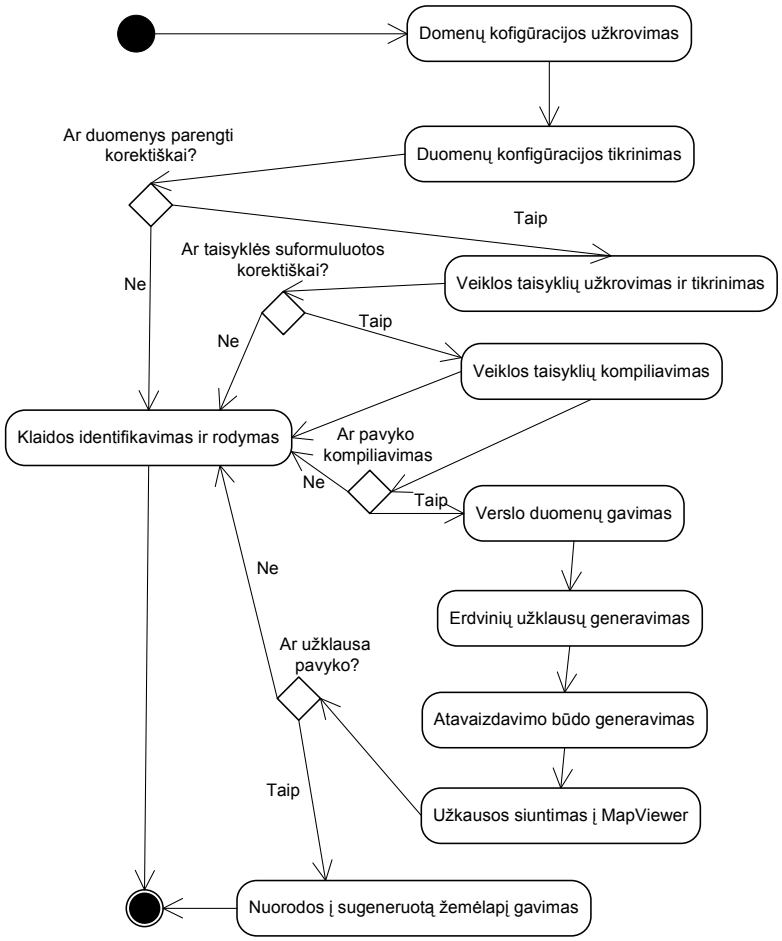


19 pav. Komponento veiklos diagrama

Pateiktoje diagramoje pavaizduota ką reikia padaryti norint, kad komponentas veiktų tinkamai. Programuotojas privalo šį komponentą „įklijuoti“ į savo programos kodą bei įterpti atitinkamas komponento iškvietimo kodo eilutes toje vietoje kurioje bus vaizduojamas žemėlapis. Tada per parametrus (konfigūruojama *.properties byla) reikia paduoti parengtas veiklos taisykles bei prisijungimo prie verslo duomenų bazės parametrus. Tada šis kodas sukompiliuojamas kartu su visa verslo taikomąja programa ir jei konfigūravimas buvo sėkmingas, gaunamas žemėlapis, jei ne – klaidos pranešimas.

4.2.KOMPONENTO ELGSENA

Komponento vidinė elgsena nusakoma žemiau pateiktoje diagramoje. Diagrama parodo kaip elgiasi komponentas nuo programuotojo nustatytų parametrų paėmimo iki sugeneruoto žemėlapio gavimo. Diagrama apima veiklos taisyklių bei verslo duomenų korektiškumo tikrinimą bei atsaką į teisingus arba neteisingus duomenis.

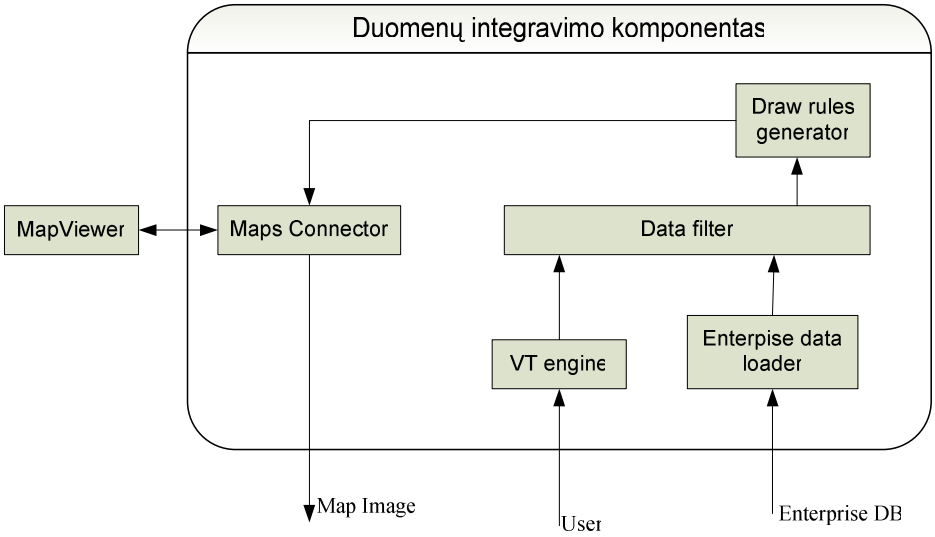


20 pav. Komponento elgsenos veiklos diagrama

Pagal pateiktą diagramą 20 pav. matome, kad pirmiausiai yra užkraunama byla, kurioje programuotojas nurodo prisijungimus prie verslo duomenų ir kitą būtiną konfigūraciją aprašytą metaduomenų skyrelyje 2.7. Sėkmingai užkrovus šią bylą patikrinama ar visi parametrai nurodyti teisingai ir ar atitinka komponento vidinius metaduomenis. Jei parametrai sukonfigūruoti teisingai, tada užkraunama veiklos taisyklių byla. Taisyklės yra patikrinamos ar jos atitinka apibrėžtą šabloną. Jei taisyklės tinkamos, tada jos kompiliuojamos ir iš jų išskiriami duomenys ir matematinės išraiškos. Vėliau pagal sukompiliuotas veiklos taisykles atrenkami tik jose minimi verslo duomenys ir sugeneruojamas komandų rinkinys pagal kurį gaunamas žemėlapio fragmentas iš MapViewer serverio pagal nurodytus stilius.

4.3. KOMPONENTO ARCHITEKTŪRA

Duomenų integravimo komponento architektūra parodyta 21 paveikslėlyje. Duomenų integravimo komponentas turi būti aprūpintas duomenų srautu iš išorinės duomenų bazės. Taip pat vartotojas turi nurodyti pagal kokias veiklos taisykles jis nori interpretuoti duomenis. Surinkus visą reikimą informaciją, įvyksta komunikacija su MapViewer‘iu ir gaunamas norimas žemėlapis.

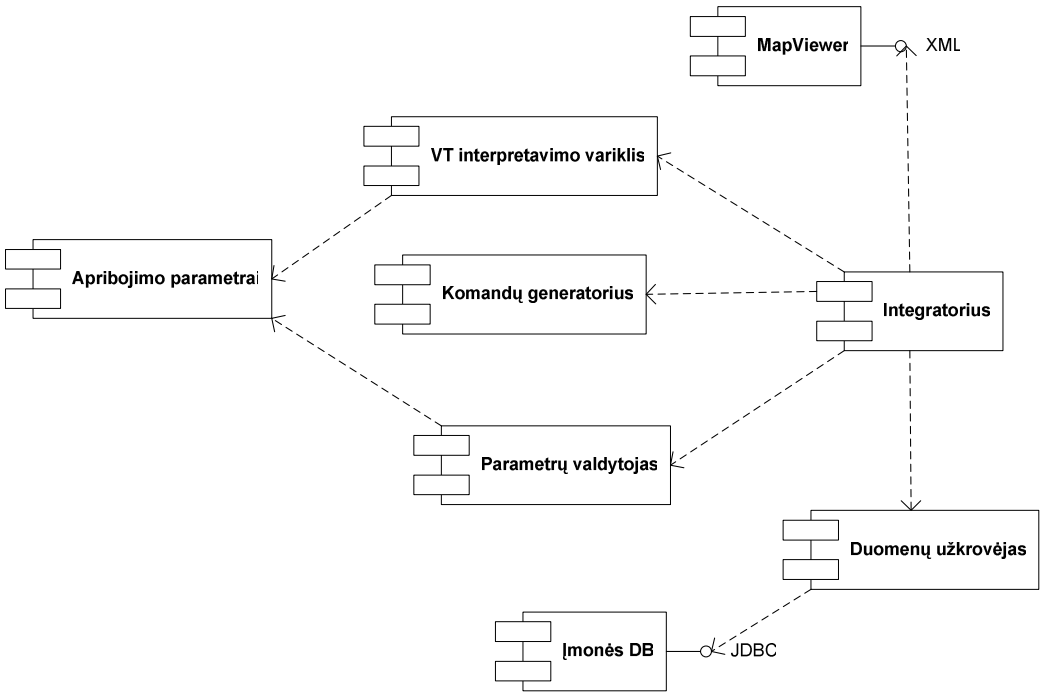


21 pav Duomenų integravimo komponento architektūra

Vartotojo taikomojoje programoje tereikia vieną kartą nurodyti veiklos taisykles pagal kurias turės būti interpretuoti duomenys. Kitaip tariant komponentas yra įprogramuojamas vartotojo taikomojoje programoje ir kiekviena kartą veiklos taisyklių nurodinėti nereikia. Kuriant kitą puslapį galima nurodyti kitokias taisykles. Taip pat pakanka vieną karta nurodyti verslo duomenų srautą ENTERPRISE DATA LOADER moduliui. Vėliau šie duomenys bus dinamiškai integruojami su erdvine informacija pagal užduotas veiklos taisykles, papildomai nieko nekonfigūruojant ir naudojantis jau nurodytais duomenų srautais.

4.4. DUOMENŲ INTEGRAVIMO KOMPONENTO STRUKTŪRA

Duomenų integravimo komponentą galima suskaidyti į mažesnes dalis kurios atlieka skirtingas funkcijas ir saugo skirtingus duomenis programiniame lygyje. Tokia komponentų diagrama pateikiama žemiau esančiame 22 paveiksle.



22 pav Duomenų integravimo komponento komponentų diagrama

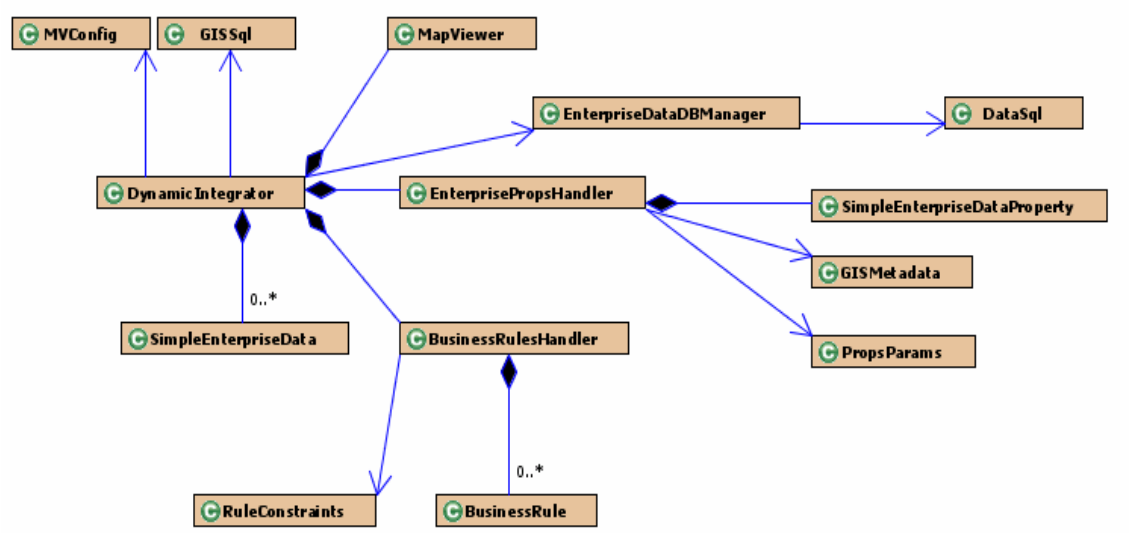
Komponentų aprašai:

- VT interpretavimo variklis – tai algoritmas kuris analizuoja kliento pateiktas veiklos taisykles, nusprendžia ar jos korektiškos ir suformuoja duomenų struktūrą kurioje saugoma išskaidytos veiklos taisyklės dalys kuriomis naudojantis formuojamos užklausos į kliento duomenų bazę ir generuojamos komandos žemėlapių gavimui.
- Duomenų užkrovėjas – tai komponentas kuris užkrauna reikalingus analizei duomenis iš įmonės duomenų bazės, priklausomai nuo duomenų bazės tipo. Šis komponentas pagal nurodytus parametrus gali prisijungti prie bet kokios duomenų bazės ir iš dinamiškai nurodomų lentelių gauti duomenis iš dinamiškai nurodomų laukų.
- Integratorius– Šis komponentas yra atsakingas už viso duomenų integravimo komponento darbą. Kviesdamas atitinkamus modulius jis žingsnis po žingsnio vykdo veiksmų sekas, kad pagal nurodytas veiklos taisykles būtų galima gauti norimą žemėlapį, žemėlapio užklausimui duomenys perduodami XML formatu.
- Apribojimo parametrai – duomenų tipai, kuriuose saugomi metaduomenų parametrai. Šie parametrai yra įprogramuoti ir negali būti keičiami vartotojo. Jie užtikrina, kad veiklos taisyklės ir komponento konfigūracinė byla būtų interpretuojamos teisingai ir kad pavyktų prisijungti prie nutolusio mapViewer komponento.
- Komandų generavimo variklis – sistemos dalis kuri iš suformuotų ir išanalizuotų duomenų struktūrų sugeneruoja žemėlapio generavimo komandas.

- MapViewer – Oracle application server komponentas generuojantis skaitmeninius žemėlapius iš erdvinės informacijos saugomos Oracle spatial duomenų bazėje.
- Įmonės DB – įmonės duomenų bazė kurioje saugoma verslo informacija.
- Parametrų valdytojas – šis komponentas užtikrina, kad programuotojo nurodyti parametrai būtų sėkmingai užkraunami ir patikrinami.

4.5.KLASIŲ DIAGRAMA

Klasių diagrama yra labiau detalizuota komponentų diagrama. Diagramoje pateikiamos klasės, kurios realizuoja duomenų integravimo komponento funkcionalumą. Kiekviena klasė realizuoja tiktai jai skirtas funkcijas stengiantis kuo mažiau priklausyti nuo kitų klasių. Tokiu būdu ateityje daromi pakeitimai vienoje komponento dalyje mažiau įtakos kitas komponento dalis.



23 pav Duomenų integravimo komponento klasių diagrama

- DynamicIntegrator – pagrindinė klasė, kuri realizuoja verslo aprašomųjų ir erdvinių duomenų integravimą ant žemėlio, naudodamasi kitomis diagramoje pateiktomis klasėmis. Šios klasės objektą programuotojas turėtų naudoti savo taikomoje programoje norėdamas naudotis komponentu. Jam tereikia sukurti objektą ir iškviesti *integrate()* metodą, kuris grąžins nuorodą į sugeneruotą žemėlapį.
- MVConfig – šioje klasėje yra įprogramuoti prisijungimo prie Oracle AS mapviewer komponento. Apie juos žino tik pats komponento kūrėjas, kadangi jis užtikrina, kad komponentas jungsis prie atitinkamo žemėlapių generavimo serverio apie kurį klientai nieko nežino.

- PropsParams – klasė aprašanti taisykles, kuriomis turi būti konfigūruojamas duomenų integravimo komponentas per išorinę bylą, kuri turi būti šakniniame taikomosios programos kataloge pavadinimu *dataConfig.properties*. Šią bylą gali laisvai keisti bet kas naudodamiesi bet kokiu tekstiniu redaktoriumi. Tokiu būdu jau sukompiliuotą ir veikiantį integravimo komponentą galima bet kada perkonfigūruoti.
- BusinessRulesHandler – ši klasė atsakinga už veiklos taisyklių užkrovimą iš šakniniame programos kataloge esančios bylos *rules.properties.* Savo žinioje ši klasė turi visas vartotojo suvestas veiklos taisykles, nesvarbu ar jos sukompiliuotos sėkmingai ar ne. Toks veiklos taisyklių užkrovimas yra lankstus, kadangi programuotojai bet kada gali pakeisti minėtos *rules.properties* bylos turinį dinaminiu būdu ir iškviesti integravimo komponentą, kuris sugeneruos žemėlapį pagal naujai sukurtas taisykles.
- BusinessRule– klasės sauganti viena veiklos taisyklę. Ji atsakinga už šios taisyklės patikrinimą ar ji atitinka apibrėžtą šabloną ir kompiliavimą jei veiklos taisyklė teisinga.
- EnterpriseDBDataManager– ši klasė leidžia prisijungti prie bet kokios dinamiškai vartotojo sukonfigūruotos duomenų bazės tam, kad būtų galima nuskaityti verslo aprašomuosius duomenis ir atvaizduoti juos žemėlapyje.
- EnterprisePropsHandler – šioje klasėje saugomi vartotojo nurodyti konfigūraciniai parametrai iš bylos *dataConfig.properties.* Pagal šiuos parametrus yra interpretuojami sakiniai veiklos taisyklėse, stengiantis išgauti duomenų objektus apie kuriuos kalbama taisyklėje.
- SimpleEnterpriseData – klasė sauganti duomenis kurie yra atrinki iš verslo duomenų bazių.
- SimpleEnterpriseDataProperty– klasė sauganti nurodytus prisijungimus prie konkrečios verslo duomenų bazės.
- DataSql – klasė turinti metodų rinkinį dinamiškam SQL užklausų generavimui verslo duomenims atrinkti.
- GisSql – klasė turinti metodų rinkinį skirtą erdvinių duomenų SQL užklausų generavimui.
- GisMetadata – įprogramuoti parametrai kurie rodo kokius erdvinius duomenis atpažįsta duomenų integravimo komponentas ir gali juos aprodyti žemėlapyje.

4.6.ŠABLONAI IR INTERPRETAVIMO VARIKLIS

Veiklos taisyklių aprašymui reiktų naudoti standartizuotą metodą. Vienas iš tokių priimtiniausių metodų būtų Business Rules Solutions RuleSpeak [11]. Šis metodas turi nestruktūrinių veiklos taisyklių aprašymo šablonais galimybę. Erdvinių ir verslo aprašomųjų duomenų integravimui nestruktūrizuotų veiklos taisyklių pilnai pakanka. Šis paketas savo sudėtyje turi pristatymo(presentation) taisyklių šabloną. Šio tipo taisyklės nurodo kaip duomenys turi būti atvaizduoti, todėl jų pilnai pakanka žemėlapių atvaizdavimui. Reikia realizuoti mechanizmą kuris turėtų interpretuoti šio tipo taisykles:

<Subject> must/should[not] BE DISPLAYED [to/on/in <media>] <display manner> [if/while <condition>]

Dalyje „<Subject>“ nurodomas erdvinio objekto tipas. Tai duomenys bendri erdviniams ir verslo duomenims. Pavyzdžiui verslo duomenys saugo kelių pavadinimus apie kuriuos sukaupę aprašomosios informacijos, o erdviniai duomenys saugo tuos pačius pavadinimus ir papildomai koordinates kur jie randasi, kad būtų galima atvaizduoti ant realaus žemėlapio.

„must/should[not] BE DISPLAYED“ yra būtina dalis šiai taisyklei.

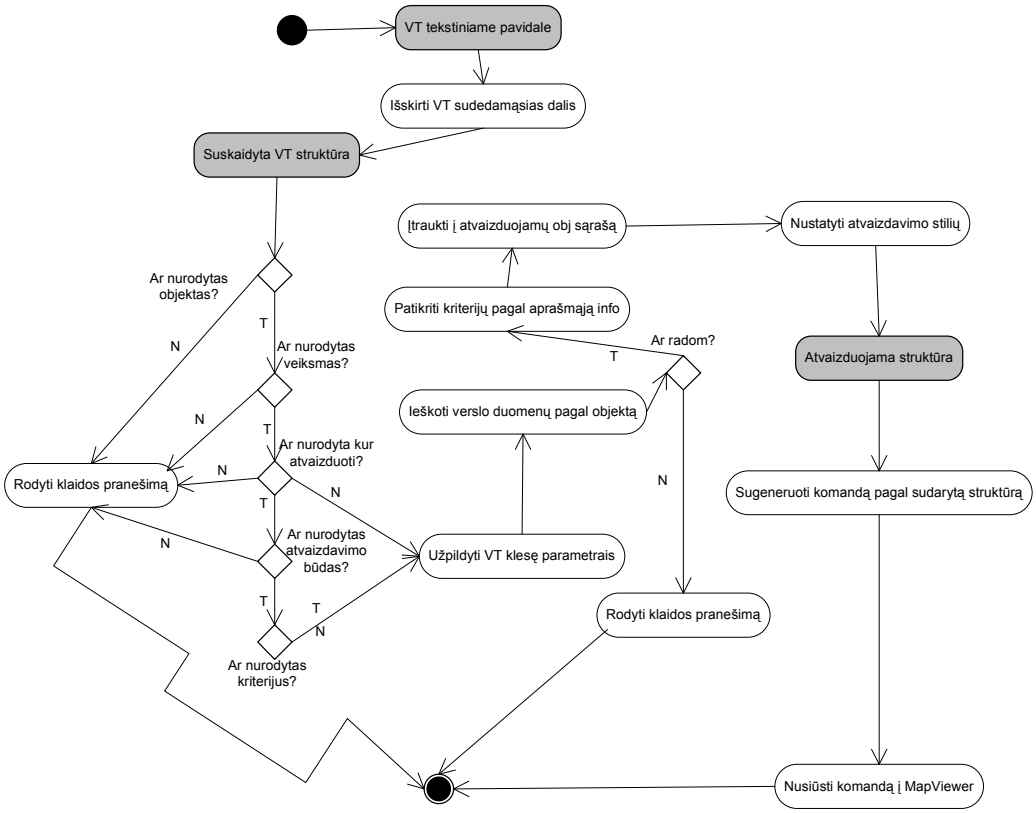
[to/on/in <media>] – reikia nurodyti kur norime atvaizduoti. Mūsų atveju atvaizduosime žemėlapyje, todėl žodelis media=MAP yra privalomas šiai taisyklei.

„<display manner>“ – nurodome kaip norime atvaizduoti. Spalvos stiliai ir t.t.

[if/while <condition>] – nurodome kada atvaizduoti. Mūsų atveju šioje sąlygoje turės būti palyginami aprašomieji verslo duomenys su taisyklėje aprašoma sąlyga ir jei ji tenkinama – atvaizduojami žemėlapyje.

Tokį šabloną atitinkanti taisyklė skambėtų taip: The roads must be displayed on the map in red if number of accidents > 10.

Veiklos taisyklių interpretavimo algoritmo schema pavaizduota 24 paveiksle.



24 pav. VT interpretavimas

Kaip matome paveiksle 24, visų pirma yra patikrinama ar veiklos taisyklė suformuluota teisingai. Pateikta tekstiniame režime ji suskaidoma ir ieškoma laukų kurie tenkina pateiktą VT šabloną. Jeigu VT suformuota neteisingai, duodamas pranešimas, kad nepavyksta suprasti veiklos taisyklės sudėties. Išrinkti reikiami laukai išsaugomi veiklos taisyklę atitinkančioje klasėje. Visi laukai patikrinami pagal nustatytus metaduomenis ir jeigu traktuojam, kad veiklos taisyklė yra suprantama, bandom ieškoti verslo duomenyse informacijos apie taisyklėje nurodomą objektą. Verslo duomenys saugo atributinę informaciją kuri naudojama atvaizdavimo sąlygoje. Tarkim nurodėme, kad norėsime atvaizduoti kelius kuriuose avarijų buvo užregistruota daugiau nei 10. Tada veiklos taisyklės klasėje yra saugomas šis palyginimo parametras ir tereikia patikrinti visus verslo duomenis apie kelius ir atrinkti tuos kurių aprašomoje informacijoje rasim, kad avarijų skaičius yra > 10. Atrinktus duomenis išsisaugome ir toje pat klasėje išsaugom atvaizdavimo stilių pagal metaduomenų apribojimus. Galiausiai iš šių surinktų duomenų suformuojam paprasčiausią komandą atitinkančia MapViewer API sintaksę ir nusiunčiam į nutolusį serverį, kuris supras šią komandą ir sugeneruos atitinkamą žemėlapio fragmentą.

4.7. VERSLO DUOMENŲ INTERPRETAVIMO PRINCIPAS

Kiekvienos įmonės verslo duomenys yra specifiniai, todėl metaduomenyse jiems nėra jokių apribojimų. Programuotojai gali nurodyti neribotą kiekį prisijungimų į skirtingas duomenų baze visame pasaulyje, kuriose bus ieškoma nurodytų aprašomųjų duomenų. Prisijungimai prie duomenų bazės, kaip ir visi metaduomenys reikalingi integravimui atlikti, nurodomi byloje dataconfig.properties. Meta duomenų konfigūravimas atliekamas tokiais žingsniais:

1. su komanda *setDB_** nurodome duomenų prisijungimo vardą, kur vietoje žvaigždutės nurodome unikalų prisijungimo vardo numerį. Tokiu būdu galima sukurti neribotą skaičių prisijungimų prie duomenų bazės.
2. su komanda *connect* nurodome prisijungimą prie duomenų bazės.
3. su komanda *table* nurodome duomenų bazės lentelę kurioje yra aprašomieji duomenys.
4. su komanda *spatField* nurodome lauką kuriame saugomas erdvinio objekto vardas kurį reikia vaizduoti žemėlapyje
5. su komanda *spatName* nurodome vardą kuriuo erdvinį objektą vadinsime veiklos taisyklėse.
6. su komanda *dataField* nurodome lentelės lauką kuriame yra aprašomieji duomenys
7. su komanda *dataName* nurodome vardą kuriuo aprašomuosius duomenis vadinsime veiklos taisyklėse.
8. su komanda *user* nurodome prisijungimo prie duomenų bazės vardą
9. su komanda *password* nurodome prisijungimo prie duomenų bazės slaptažodį.
10. su komanda *spatMeta* nurodome erdvinių metaduomenų tipą, t.y. nurodome duomenų integravimo komponentui kokių erdvinių duomenų reikia ieškoti GIS serveryje.
11. su komanda dbType nurodom duomenų bazės tipą prie kurios jungsimės.

Pavyzdinis prisijungimas prie duomenų bazės galėtų atrodyti taip:

```
setDB_1=keliai1
setDB_2=keliai2

keliai1.connect=jdbc:mysql://localhost:3306/accidents_on_roads1
keliai1.table=accidents_in_vln
keliai1.spatField=road_name
keliai1.spatName=roads
keliai1.dataField=value
keliai1.dataName=number of accidents
keliai1.user=root
```

```
keliai1.password=Gerietis
keliai1.spatMeta=roads
keliai1.dbType=mySql

keliai2.connect=jdbc:mysql://localhost:3306/accidents_on_roads_report
keliai2.table=accidents
keliai2.spatField=street_name
keliai2.spatName=roads
keliai2.dataField=accidents_numb
keliai2.dataName=number of accidents
keliai2.user=root
keliai2.password=Gerietis
keliai2.spatMeta=roads
keliai2.dbType=mySql
```

Matome, kad galima nurodyti neribotą kiekį prisijungimų prie duomenų bazių. Tokius prisijungimus pakanka nurodyti tik vieną kartą ir vėliau papildomai nieko nekonfigūruojant bus galima veiklos taisyklių pagalba matyti įmonių aprašomuosius duomenis įvairiais pjūviais.

5. DINAMIŠKOS GIS SISTEMOS KOMPONENTŲ DERINIMAS

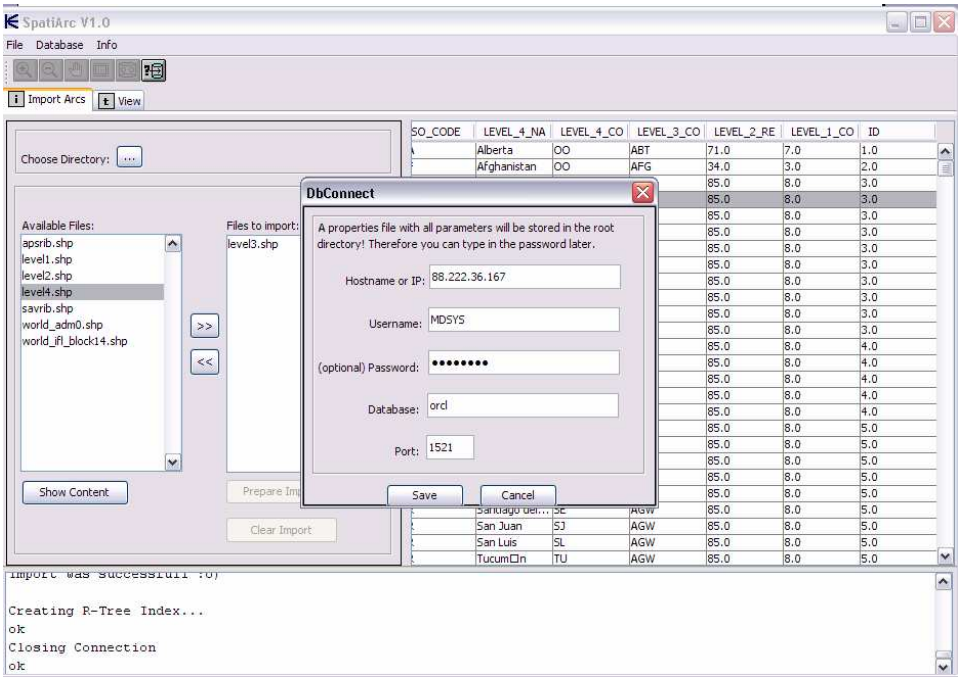
Dinamiškos sistemos realizacija neapsiriboja vien duomenų integravimo komponento sukūrimu. Sistemos realizaciją galima suskaidyti į tris pagrindinius etapus, kurie yra svarbūs ir sistema negali funkcionuoti be jų.

- Erdvinės informacijos įkėlimas į Oracle spatial DB
- MapViewer diegimas ir konfigūravimas
- Duomenų integravimo komponento realizacija

5.1. ERDVINIŲ DUOMENŲ PARENGIMAS

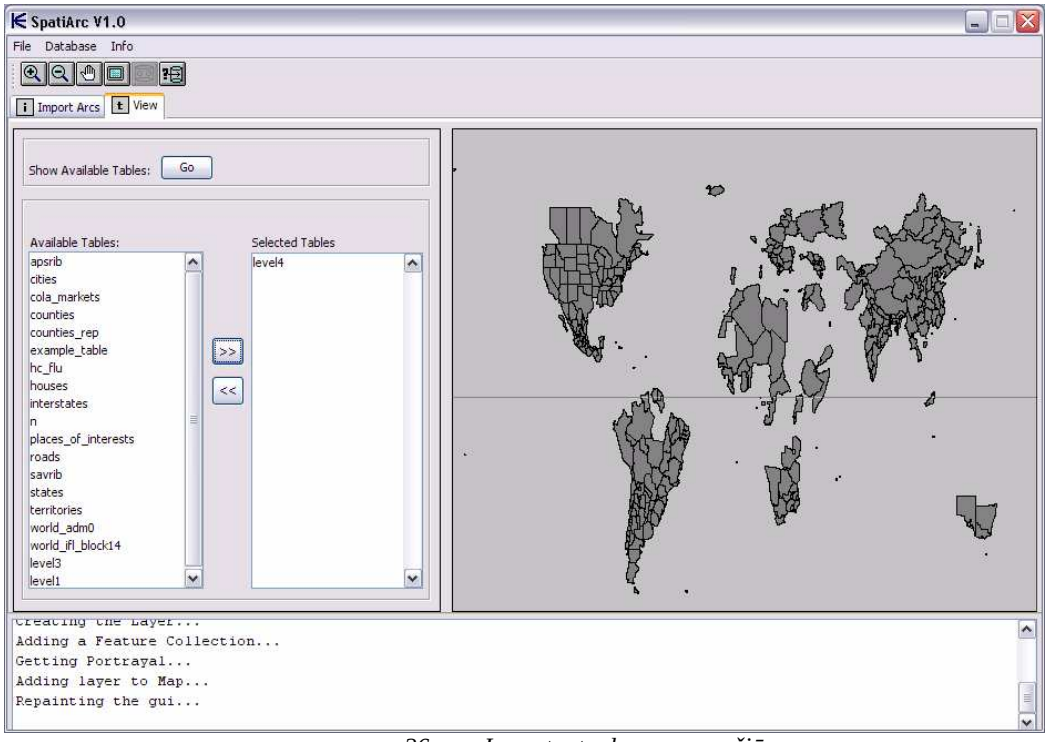
5.1.1. ESAMŲ DUOMENŲ IMPORTAVIMAS

Norėdami parengti pavyzdinę erdvinių duomenų bazę, importuosime jau egzistuojančius erdvinius duomenis. Pasinaudosime jau egzistuojančiais erdviniais duomenimis, kadangi jų kūrimas rankiniu būdu taptų ilgu ir nereikalingu procesu. Tuo tikslu panaudosime paprasčiausia atviro kodo internete laisvai prieinamą SpatiArc V1.0 kuri importuoja erdvinę informaciją į Oracle duomenų bazę iš esri shape formato. Lygiai taip pat internete randame laisvai pasiekiamų nemokamų erdvinių duomenų minėtame formate ir importuojame juos į Oracle spatial duomenų bazę. Importuojami duomenys atitinka pasaulio žemynų ir šalių ribas. Importuojamų bylų pavadinimai: level4.shp, level3.shp, level2.shp, level1.shp.



25 pav. Erdvinių duomenų importavimas

Paveikslėlyje 25 matome kaip nurodome prisijungimo parametrus prie egzistuojančios Oracle duomenų bazės ir pasirinkę norimą *.shp bylą importuojame ją į duomenų bazę. Tokiu būdu automatiškai importuojame didelį informacijos kiekį kas rankiniu būdu būtų praktiškai neįmanoma. Taigi dabar turime pavyzdinių duomenų su kuriais galime atlikti analizę. Šie duomenys apima realiai egzistuojančias teritorijas pasaulyje. Žemiau pateiktame paveikslėlyje 26 parodoma kaip atrodo importuoti duomenys iš bylos level3.shp. Analogiškai importuojamos ir likusios bylos.

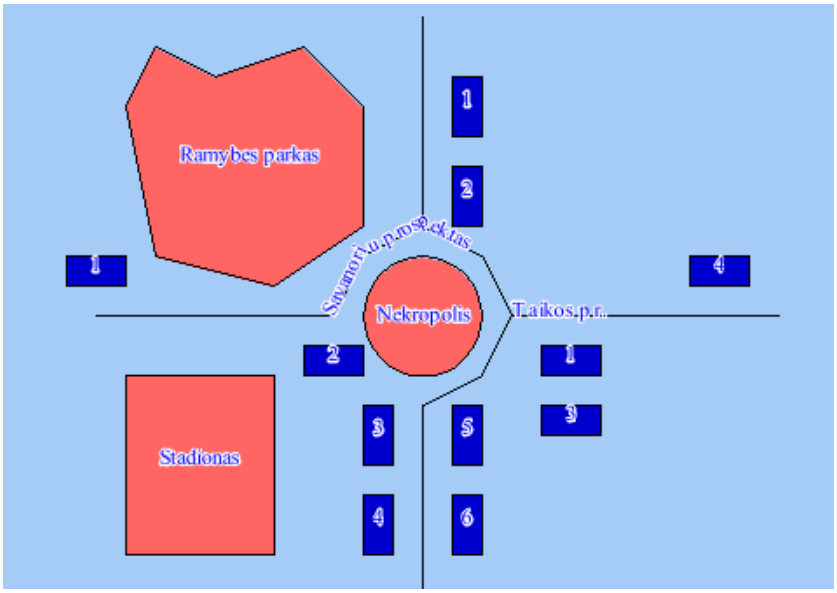


26 pav. Importuotų duomenų peržiūra

Patiktame paveiksle 26 parodytas žemėlapis, kuriame yra atvaizduoti mūsų importuoti duomenys, kuriais galės naudotis duomenų integravimo komponentas. Lango dešinėje pusėje matome, kad erdviniai duomenys buvo importuoti sėkmingai ir iš jų galima gauti žemėlapį.

5.1.2. NAUJŲ DUOMENŲ KŪRIMAS

Šalia automatiškai importuotų erdvinių duomenų, sukursime keletą rankiniu būdu. T.y. sukursime gatvių sluoksnį, namų ir lankytinų vietovių. Sukursime tik keletą pavyzdinių šių objektų egzempliorių, kadangi erdvinių duomenų kūrimo procesas yra ilgas ir mechaniškas darbas. SQL komandos reikalingos šiems objektams sukurti pateiktos priede 3. Žemiau esančiame paveiksle 27 pateikiamas žemėlapis, kurį sudaro keletas rankiniu būdu sukurtų erdvinių objektų.



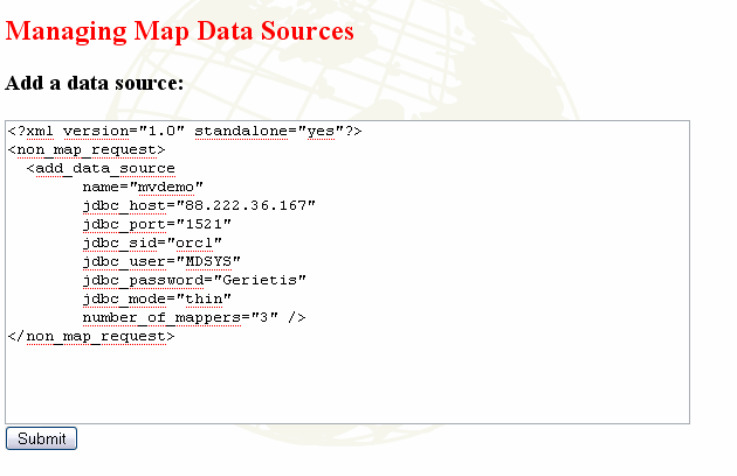
27 pav. Rankiniu būdu sukurti erdviniai duomenys

Paveiksle 27 matome tris atskirus sluoksnius. Matome mėlynai nuspalvintus namus, juodai – kelius, raudonai – lankytinas vietas. Rankomis sukurtų erdvinių duomenų nėra daug, tačiau juos galėsim naudoti duomenų integravimo komponente.

5.2. MAPVIEWER KONFIGŪRAVIMAS

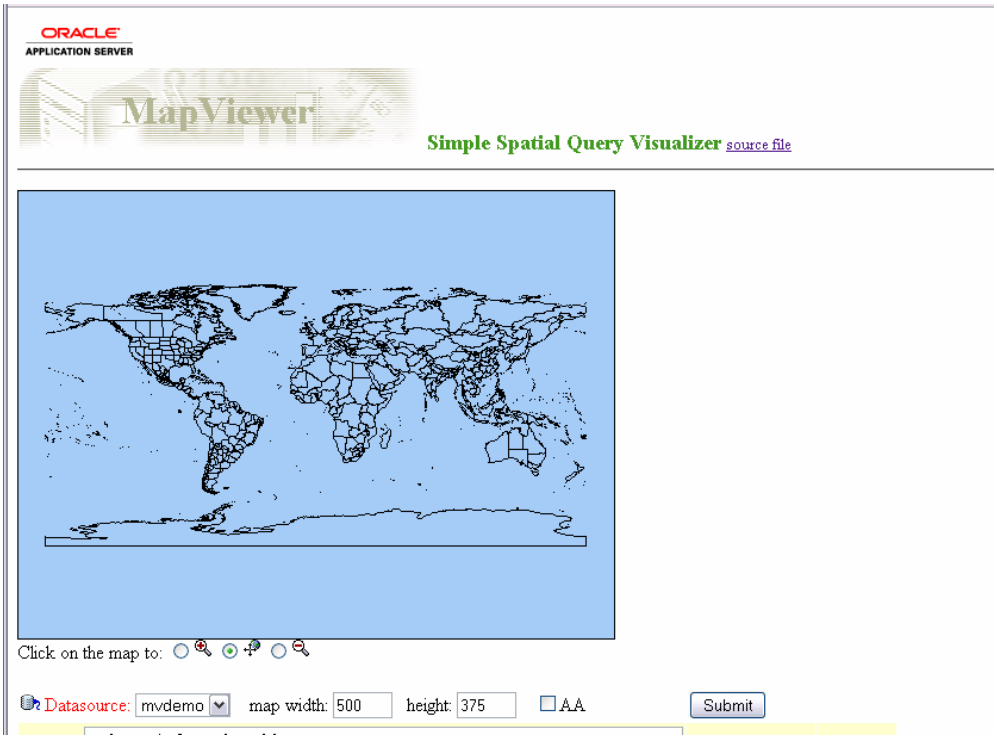
MapViewer yra serverio pusės komponentas, kuris sukuria žemėlapius skaitydamas tinkamos duomenų bazės duomenis ir grąžinantis klientui žemėlapius tinkamu formatu. Kiekvieno žemėlapio formavimas yra atliekamas naudojantis vienu ar daugiau sluoksnių arba temų. Kiekviena tema apibūdina grupę erdvinių objektų tokių kaip keliai, upes ir t.t. Šie objektai yra vaizduojami atitinkamais stiliais.

Norint naudoti MapViewer reikia jį įdiegti į Oracle AS serverį, tada tinkamai sukonfigūruoti ir nurodyti erdvinių duomenų srautą. Minėtą srautą suformuosime pavyzdiniams duomenims. Tokiu būdu įsitikinsime kad duomenys tikrai realiai egzistuoja ir galime juos pasiekti žemėlapių formavimui.



28 pav. Duomenų srauto nustatymas

Pateiktame paveiksle 28 matome kaip nurodomas duomenų srautas į duomenų bazę. Šis duomenų srautas egzistuoja nutolusiame serveryje kuriame yra įdiegtas Oracle AS. Duomenų srautas buvo sukurtas tam, kad per jį būtų galima perduoti/gauti duomenis iš MapViewer API kuris yra naudojamas šio darbo metu sukurtame duomenų integravimo komponente. Duomenų integravimo komponente yra klasė MVConfig.class kurioje yra įprogramuoti duomenys apie šį duomenų srautą. Tokiu būdu klientai paprasčiausiai gali naudotis duomenų integravimo komponentu visiškai nesirūpindami dėl to koks duomenų srautas yra sukurtas GIS serveryje ir koks duomenų srautas sukonfigūruotas duomenų integravimo komponente, kadangi natūralu kad ir vienoje ir kitoje pusėje naudojamasi tuo pačiu duomenų srautu. Norėdami patikrinti ar MapViewer veikia tinkamai, jo pagalba užkrausime ankstesniame punkte importuotus erdvinius duomenis pasinaudodami demonstracine Oracle sukurta MapViewer testavimo taikomąja programa.

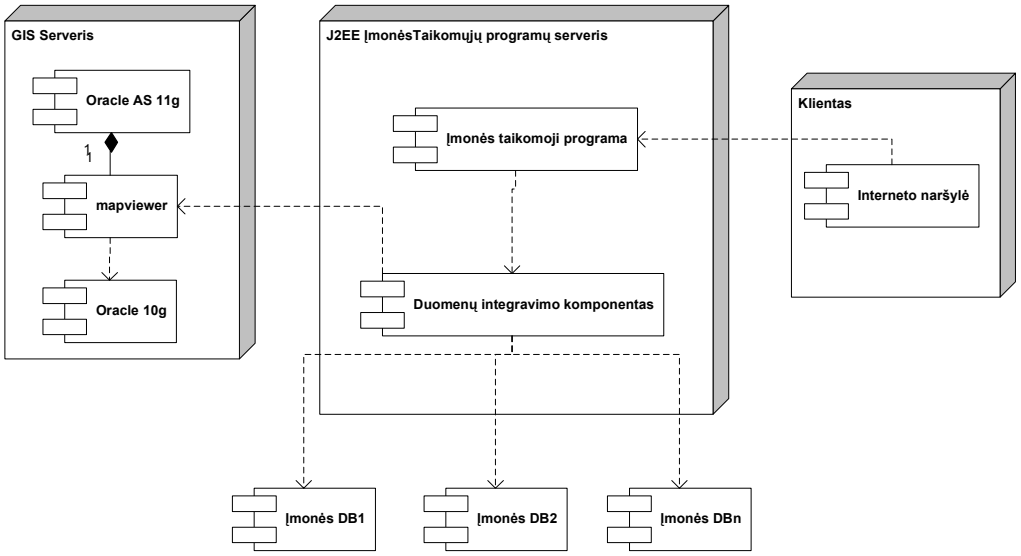


29 pav. Sugeneruotas žemėlapis

Pateiktame paveiksle 29 matome viso pasaulio žemėlapį ir sukurtą minimalų funkcionalumą su žemėlapiu. Galima jį judinti artinti bei tolinti. Taip pat matome, kad žemėlapis yra gaunamas per anksčiau sukonfigūruotą duomenų srautą „mvdemo“ Tokiu būdu įsitikiname, kad MapViewer komponentas veikia ir sukonfigūruotas teisingai, todėl galėsime juo naudotis dinamiškam žemėlapių generavimui.

5.3. SISTEMOS DIEGIMAS

Norint, kad dinamiška GIS sistema būtų galima realizuoti, reikia, kad egzistuotų šie komponentai: Oracle 10g duomenų bazė, Oracle AS 11g taikomųjų programų serveris, MapViewer J2EE komponentas, sukurtas duomenų integravimo komponentas, įmonės taikomoji programa kurioje yra naudojamas minėtas komponentas bei įmonės duomenų bazės, kuriose yra saugomi verslo aprašomieji duomenys.



30 pav. Dinamiškos GIS sistemos diegimo diagrama

Paveikslėlyje 30 pateikta dinamiškos GIS sistemos diegimo diagrama. GIS serveryje turi būti suinstaliuota Oracle taikomosios programos. Oracle AS 11g yra taikomųjų programų serveris, kuriame turi būti patalpintas mapViewer komponentas. Šis komponentas gali egzistuoti tik Oracle AS11g aplinkoje ir negali veikti kaip laisva nepriklausoma taikomoji programa. Oracle 10g yra duomenų bazė su Spatial funkcijomis ir joje yra laikomi erdviniai duomenys. MapViewer komponentas per nurodytą duomenų srautą naudojasi minėtais duomenimis tam, kad galėtų sugeneruoti žemėlapį. J2EE įmonės taikomųjų programų serveryje turi būti paleista įmonės taikomoji programa parašyta java kalba ir kurioje yra įdėtas šio magistrinio darbo metu sukurtas duomenų integravimo komponentas. Šis komponentas naudojasi duomenimis iš įmonės duomenų bazių ir atvaizduoja juos žemėlapyje. Klientas norėdamas naudotis įmonės sukurtomis paslaugomis turi turėti interneto naršyklę per kurią jis pasiekia įmonės taikomąją programą.

6. DINAMIŠKOS GIS SISTEMOS TESTAVIMAS

Siekiant ištestuoti duomenų integravimo komponentą, buvo sukurta testinė sąsaja, kurioje galima įvesti veiklos taisykles ir matyti pagal jas sugeneruotą žemėlapį. Testinė sąsaja yra internetinis puslapis, kuris demonstruoja kaip duomenų integravimo komponentas gali buti pakartotinai panaudotas ir sukonfigūruotas įmonių taikomosiose programose, kadangi pačią testavimo sąsają galima traktuoti kaip įmonės taikomąją programą.

6.1. KOMPONENTO TESTINĖ KONFIGŪRACIJA

Sukurtas duomenų integravimo komponentas yra universalus ir pakartotinai panaudojamas įmonės taikomosiose programose. Duomenų integravimo komponentas yra sukompiliuotas į *.jar bylą, todėl kuriant įmonėms taikomąsias programas, reikia tik įklijuoti šią bylą į įmonės taikomosios programos projektą bei sukūrus *DynamicIntegrator* tipo objektą iškvieti metodą *integrate()*. Tokiu būdu bus grąžintas žemėlapio paveikslėlio adresas. Prieš iškviečiant *integrate()* metodą, reikia užtikrinti, kad komponentas yra tinkamai sukonfigūruotas ir jam nurodytos veiklos taisyklės. Komponento konfigūracijai yra naudojamos dvi bylos, kurias galima sukonfigūruoti vieną karta prieš kompiliuojant įmonės taikomąją programą arba keisti konfigūraciją dinamiškai, įmonės taikomųjų uždavinių vykdymo metu. Konfigūravimo bylos turi būti taikomosios programos šakniniame kodo kataloge ir pavadintos sekančiai:

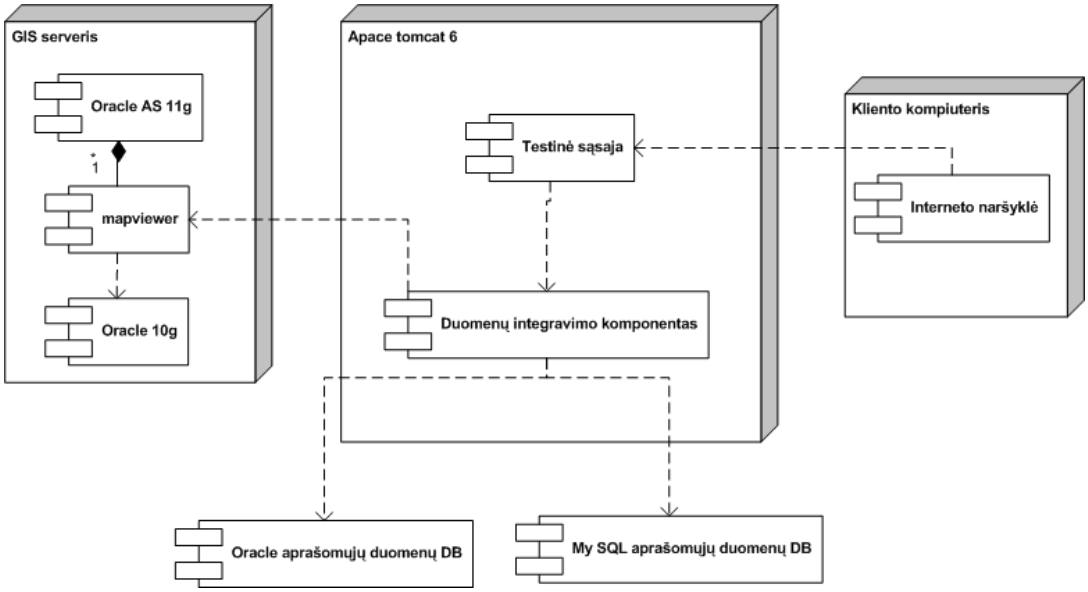
- *dataconfig.properties* – šioje byloje aprašomi prisijungimai prie įmonės duomenų bazių bei aprašomieji duomenys kuriuos norėsime matyti žemėlapyje. Šią bylą galima laisvai keisti įmonės taikomųjų uždavinių vykdymo metu arba sukonfigūruoti viena kartą ir vėliau naudoti tą pačią konfigūraciją. Sistemos testavimo atveju šią bylą sukonfigūruosime vieną karta ir vėliau rezultatus gausim nebesirūpindami duomenų konfigūravimu. Šios bylos pavyzdys yra pateiktas ketvirtame priede.
- *rules.properties* – šioje byloje turi būti surašytos veiklos taisyklės, kurios bus kompiliuojamos iškvietus metodą *integrate()*. Taisykles galima nurodyti vieną kartą, siekiant visada gauti ta pati žemėlapį, arba jas keisti dinamiškai įmonės taikomųjų uždavinių vykdymo metu. Sistemos testavimo atveju, sukūrėme testinę sąsają, kuri imituoja įmonės taikomuosius uždavinius ir keičia veiklos taisykles dinamiškai. Testinėje sąsajoje vartotojas gali bet kada įrašyti veiklos taisykles pagal kurias nori matyti žemėlapį. Tada testinės sąsajos taikomieji uždaviniai įrašo šias vartotojo įvestas taisykles į *rules.properties* bylą ir iškvietus metodą *integrate()* sugeneruoja žemėlapį.

6.2.TESTINĖS SISTEMOS PROTOTIPO DIEGIMAS

Testavimo metu atlikome visos sistemos testavimą. Importuotiems erdviniams duomenims atsitiktinai sugeneravome aprašomuosius įmonės duomenis kuriuos patalpinome i skirtingas įmonės duomenų bazes. Tokiu būdu siekiama parodyti, kad duomenų integravimo komponentui nėra svarbu iš kur yra duomenys ir kiek jų įmonė turi. Atsitiktinai sugeneravome tokius aprašomuosius duomenis, kurios naudojame testavime:

- Žemynuose esančių šalių skaičius
- Žemynuose esančių gyventojų skaičius
- Žemynuose esančių religijų skaičius
- Žemynų laiko juostas
- Šalių prezidentų lytį
- Laiką kada paskutinį kartą šalyse buvo išrinktas prezidentas
- Šalyje vyraujančią religiją

Žemiau pateiktame paveiksle 31 yra pavaizduota sistemos testavimo diegimo diagrama.



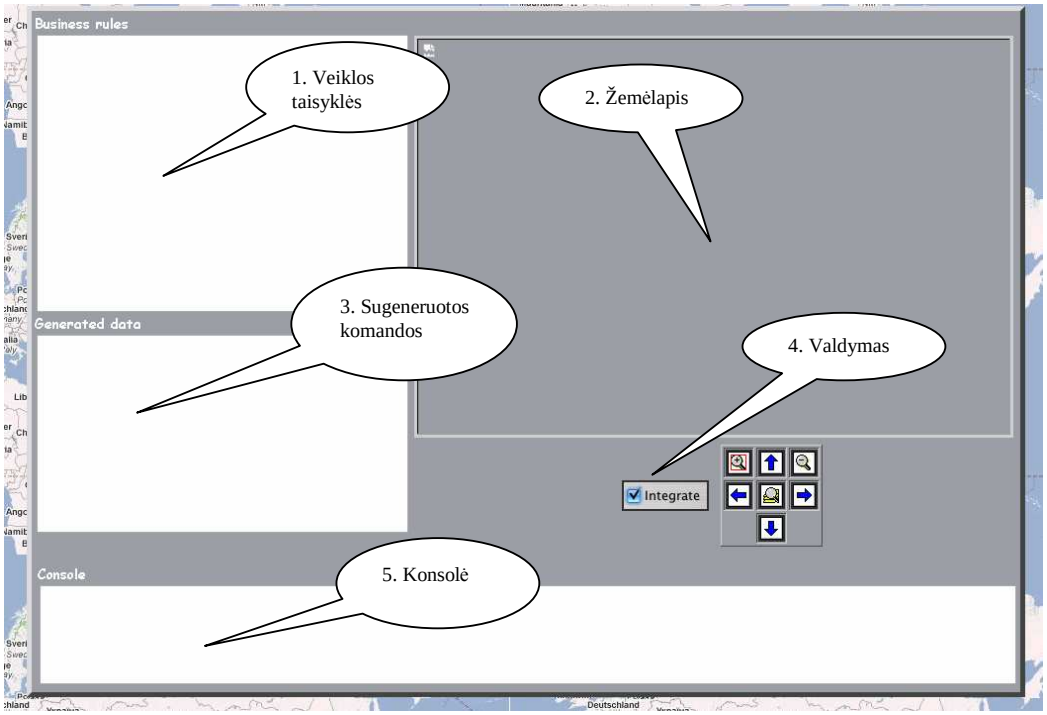
31 pav Dinamiškos GIS sistemos testavimo diegimo diagrama

Sistemos struktūra yra tokia pati kaip ir aptarta 30 pav. Papildomai galima patikslinti įmonės taikomųjų paslaugų dalį. Testavimo atveju įmonės taikomąja programa galima laikyti testavimo sąsają, kuri yra patalpinta į tomcat6 serverio konteinerį. Testavimo sąsajoje yra įdėtas duomenų integravimo komponentas kuris sukonfigūruotas prisijungimui prie dviejų

įmonės išorinių duomenų bazių: Oracle ir mySql. Komponentas ima duomenis iš šių duomenų bazių ir atsižvelgdamas į veiklos taisykles vaizduoja juos žemėlapyje be programuotojo įsikišimo.

6.3.TESTAVIMO SĄSAJOS LANGAS

Komponento reakciją į veiklos taisykles ištestuosime pasinaudodami sukurta testavimo sąsaja. Šios sąsajos langas yra parodytas žemiau esančiame paveiksle 32.



32 pav. Dinamiškos GIS sistemos testavimo lango pavyzdys

1 – Šioje srityje galima suvesti veiklos taisykles pagal kurias norime matyti žemėlapį.

2 – Šioje srityje matomas žemėlapis sugeneruotas pagal veiklos taisykles.

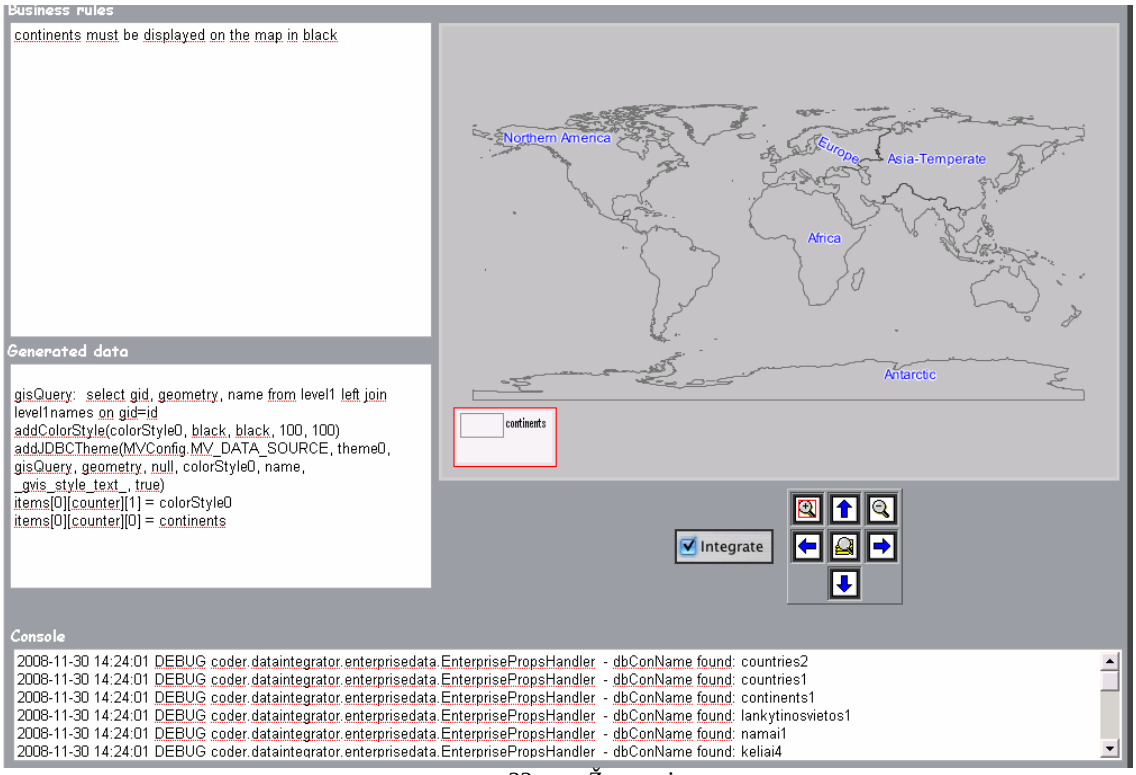
3 – Šioje srityje yra išvedamos komandos kurias duomenų integravimo komponentas generuoja automatiškai be programuotojo pagalbos.

4 – Šioje srityje yra bazinės žemėlapio valdymo funkcijos ir mygtukas „Integrate“ kurį paspaudus veiklos taisyklės surašomos į bylą rules.properties ir iškviečiamas duomenų integravimo komponento *integrate()* metodas.

5 – Šioje srityje matome informacija apie sėkmingai sukompiliuotas veiklos taisykles bei klaidas veiklos taisyklėse arba prisijungimo prie duomenų bazių konfigūravimo byloje.

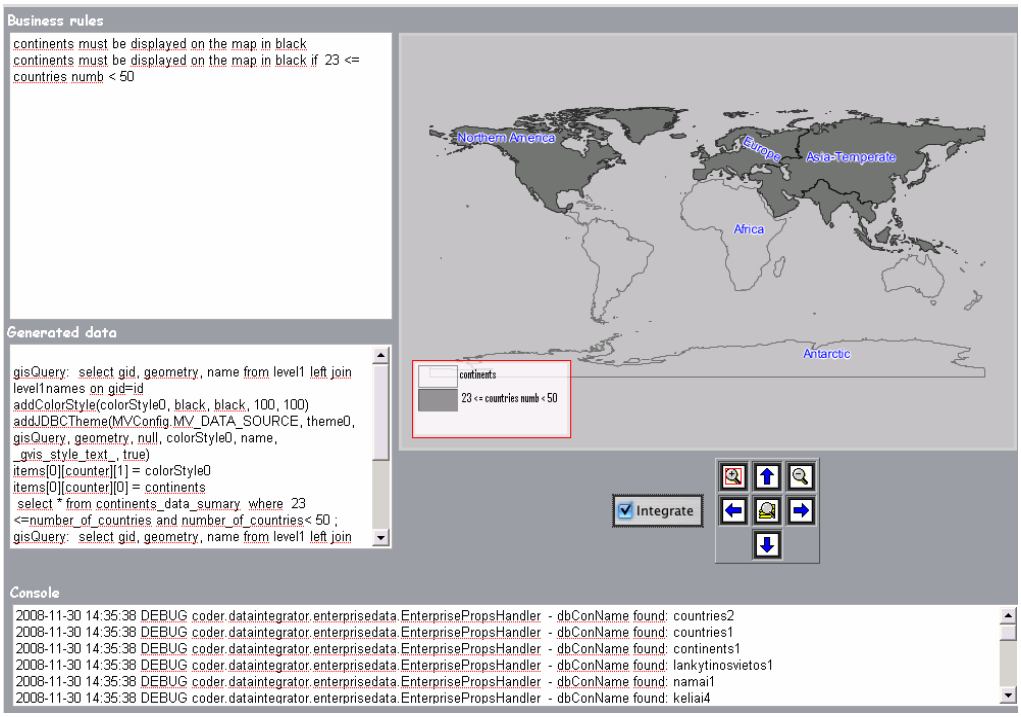
6.4. VEIKLOS TAISYKLIŲ TESTAVIMAS

Atlikdami testavimą, naudodami veiklos taisykles susiesime testavimui sugeneruotą aprašomąją informaciją su importuota į Oracle spatial erdvine informacija. Visų pirma duokime komandą duomenų integravimo komponentui, kad jis žemėlapyje parodytų visus žemynus juodais kontūrais. Tam parašome tokią veiklos taisyklę: *continents must be displayed on the map in black*. Rezultatus matome žemiau pateiktame paveiksle 33.



33 pav. Žemynai

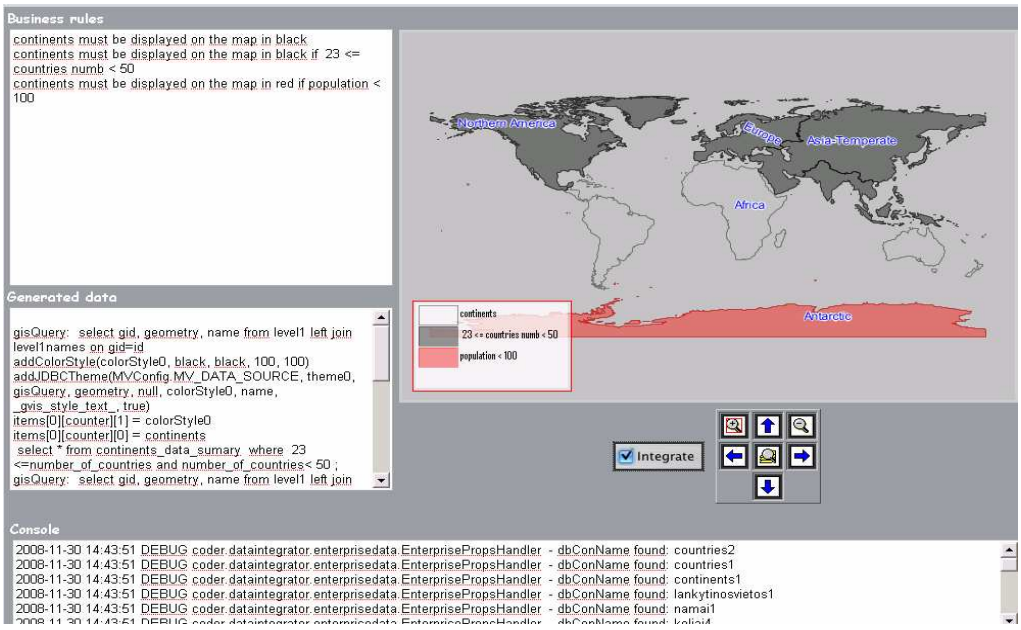
Dabar panaudokime aprašomuosius duomenis apie valstybių skaičių žemyne ir žemėlapyje pavaizduokime valstybes juodai, jei valstybių skaičius juose yra daugiau nei 23, bet mažiau nei 50. Tuo tikslu parašome tokią veiklos taisyklę: *continents must be displayed on the map in black if 23 <= countries numb < 50*. Rezultatai pateikiami žemiau esančiame paveiksle 34.



34 pav. Juodai nuspalvinti žemynai

Paveiksle 34 matome, kad žemynai kurių įmonės aprašomieji duomenys atitiko užduotą kriterijų, buvo nuspalvinti juodai.

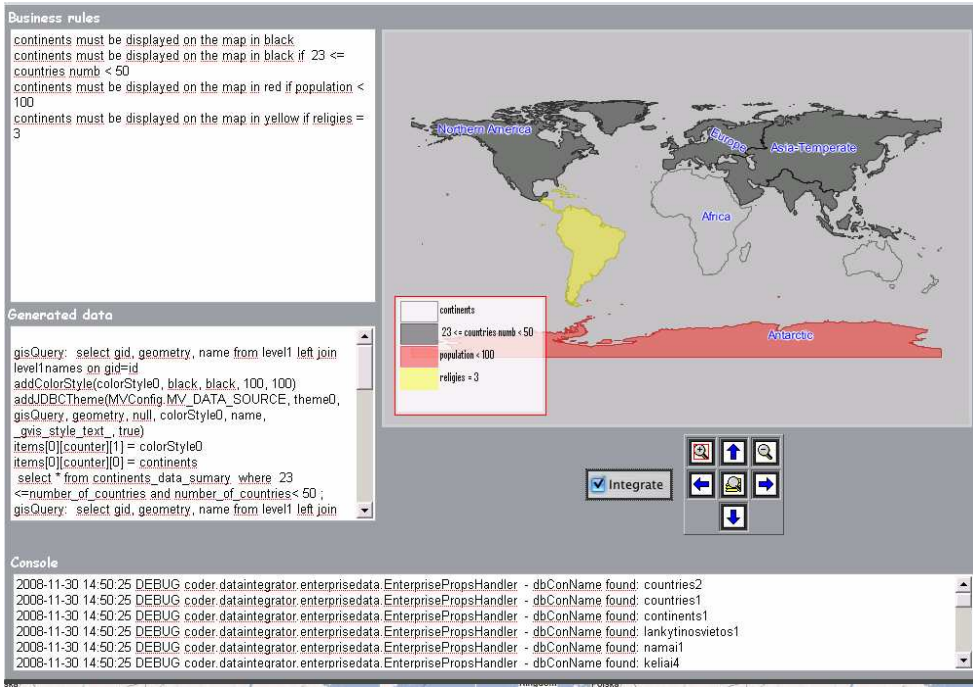
Dabar panaudokime aprašomuosius duomenis apie gyventojų skaičių žemynuose ir raudonai nuspalvinkim tuos žemynus, kur gyventojų yra mažiau nei 100. Tuo tikslu parašome veiklos taisyklę: *continents must be displayed on the map in red if population < 100*. Rezultatai pateikiami 35 paveikslėlyje.



35 pav. Žemynai kur gyventojų < 100.

Paveikslėlyje 35 matome, kad tik Antarktidoje gyventojų yra mažiau nei 100 todėl ji rodoma raudonai.

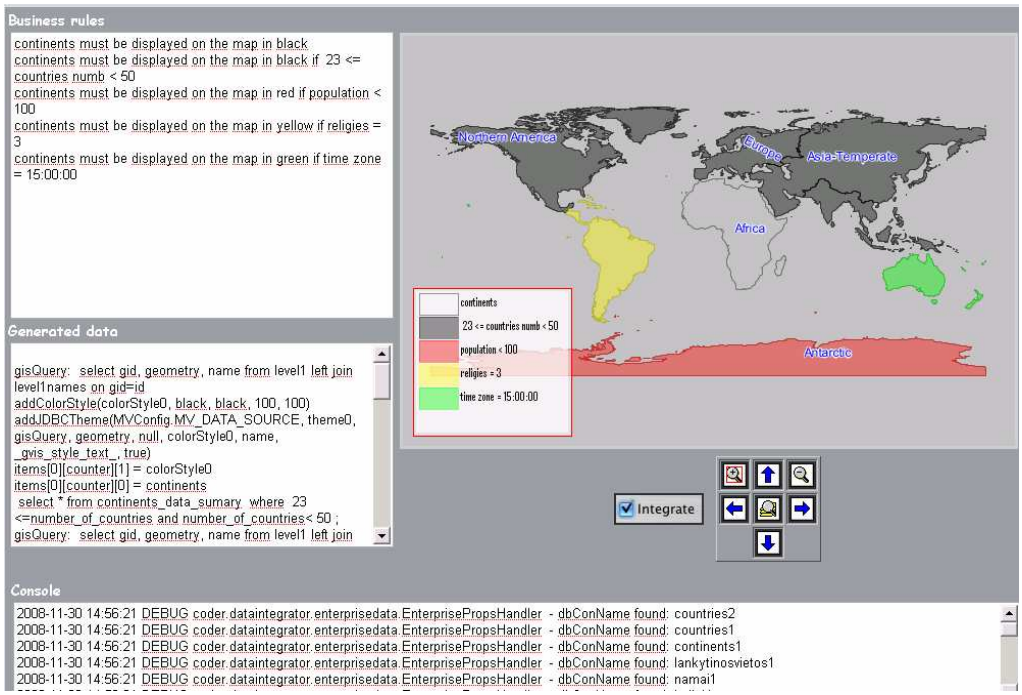
Siekdami panaudoti aprašomuosius duomenis apie skirtingų religijų skaičių žemynuose, parašykime veiklos taisyklę, kuri geltonai nuspalvins žemynus, kuriuose yra paplitę lygiai 3 skirtingų rūšių religijos: *continents must be displayed on the map in yellow if religies = 3*. Rezultatai pateikiami 36 paveiksle.



36 pav Žemynai kur skirtingų religijų yra 3

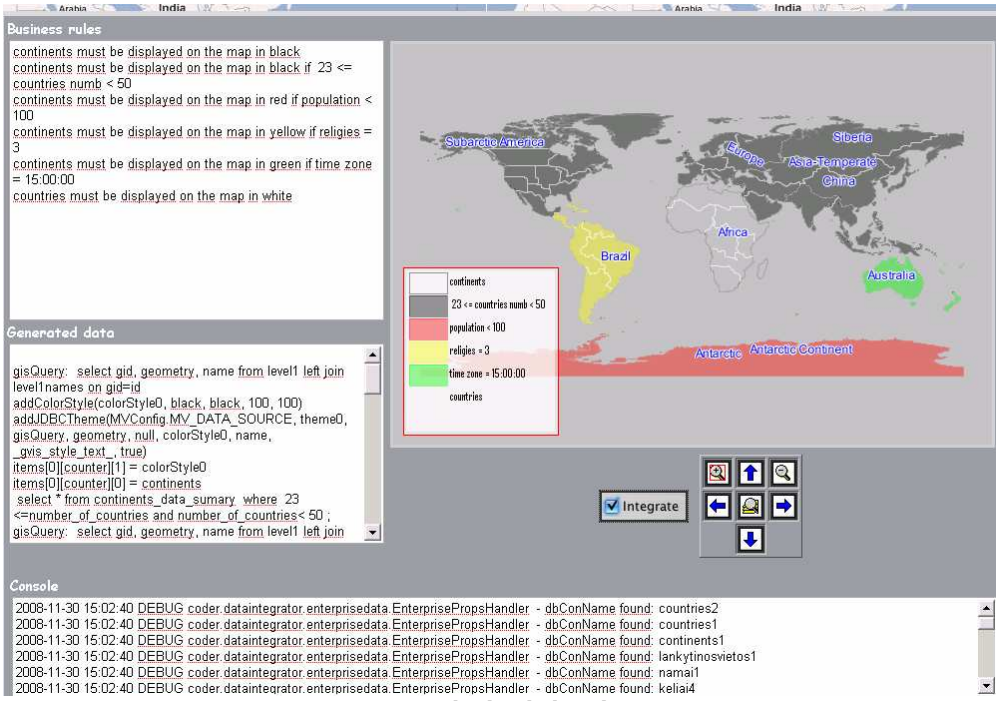
Paveikslėlyje 36 matome, kad lygiai 3 skirtingas religijas turi pietų Amerika, todėl ji rodoma geltonai.

Siekdami išnaudoti aprašomuosius duomenis apie laiko juostas žemynuose, nuspalvinsime žemynus žaliai, jei jie yra 15:00:00 laiko juostoje. Tai yra išgalvoti duomenys ir realių laiko juostų jie žinoma neatitinka. Parašome veiklos taisyklę: *continents must be displayed on the map in green if time zone = 15:00:00*. Rezultatai pateikiami 37 paveikslėlyje.



37 pav žemynai esantys 15:00:00 laiko juostoje rodomi žaliai

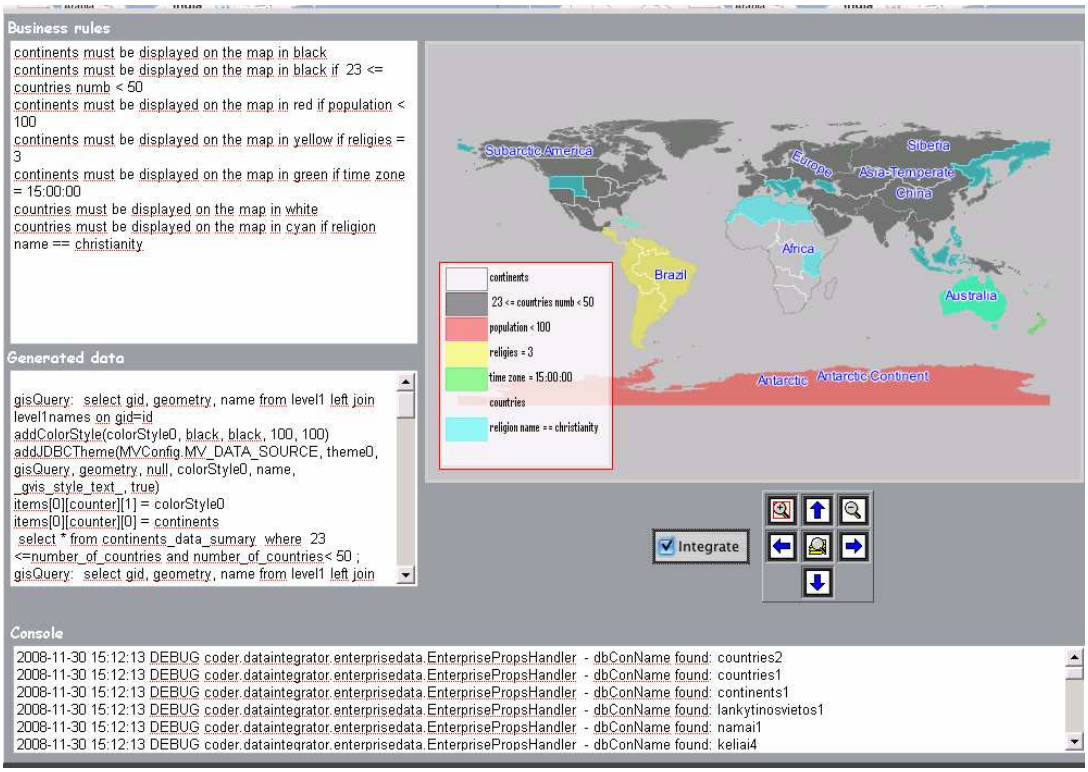
Dabar šalia sugeneruoto žemėlapio parodykime valstybių kontūrus baltai. Tokiu būdu matysime anksčiau sugeneruotą žemėlapį, tačiau jį dar labiau detalizuosime valstybių kontūrais. Tuo tikslu parašome veiklos taisyklę: *countries must be displayed on the map in white*.



38 pav Valstybės baltais kontūrais

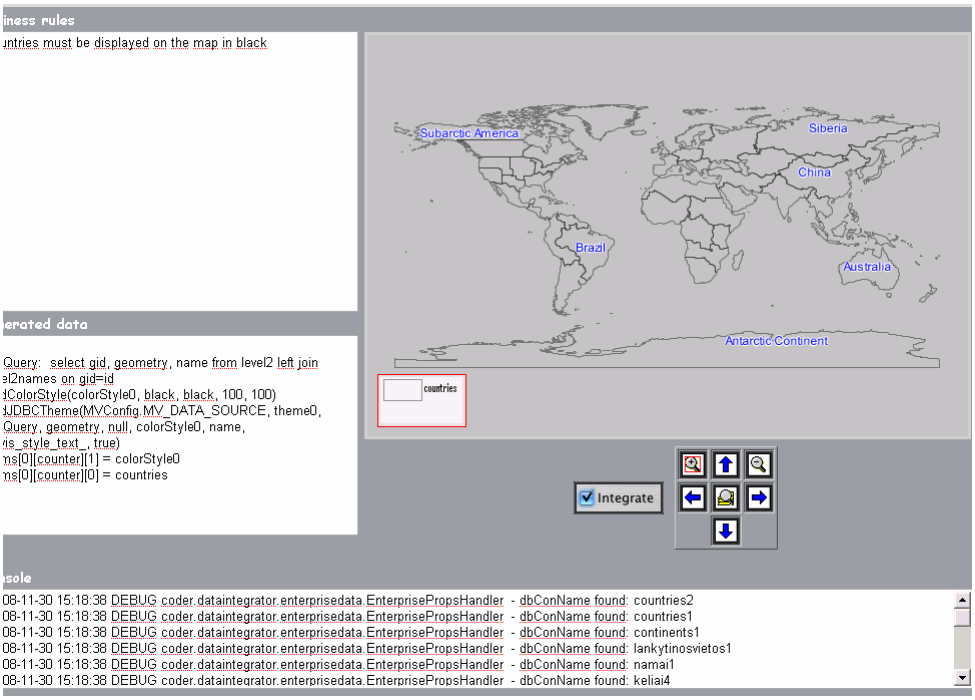
Paveikslėlyje 38 matome detalesnį žemėlapį, kuriame valstybių kontūrai yra rodomi baltai.

Dabar panaudokime aprašomuosius duomenis apie vyraujančias religijas valstybėse ir valstybes kuriose vyraujanti religija yra krikščionybė nuspalvinkime melsvai. Tuo tikslu parašome veiklos taisyklę: *countries must be displayed on the map in cyan if religion name == christianity.*



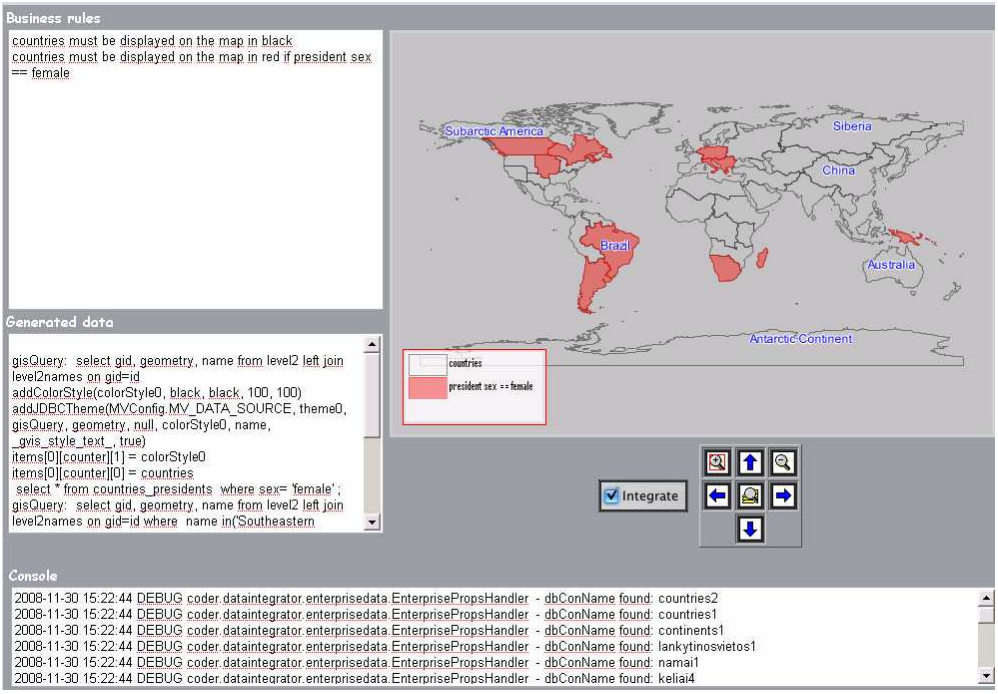
39 pav Valstybės kuriose vyraujanti religija krikščionybė nuspalvintos melsvai

Dabar pašalinkime visas spalvas ir žemėlapyje palikome tik juodus valstybių kontūrus. Tam paliekame tik viena veiklos taisyklę: *countries must be displayed on the map in black*. Rezultatai matomi žemiau esančiame 40 paveiksle.



40 pav Valstybės su juodais kontūrais

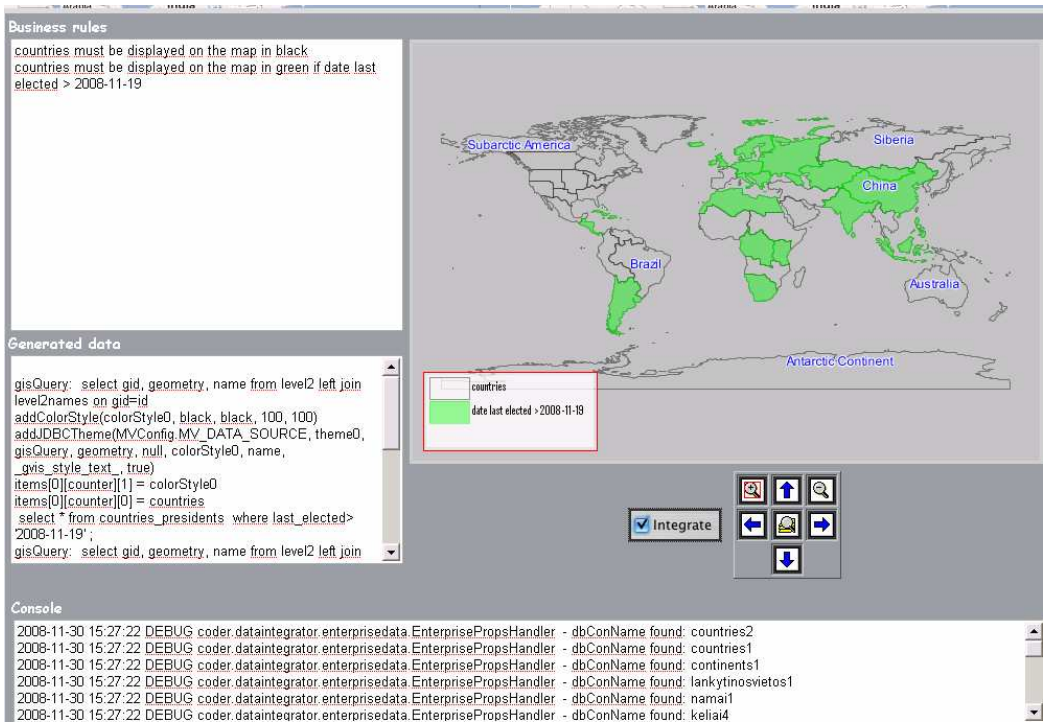
Panaudokime aprašomuosius duomenis apie valstybes valdančių prezidentų lytį. Tuo tikslu raudonai nuspalvinime visas valstybes kurias valdo moterys. Tam parašome veiklos taisyklę: *countries must be displayed on the map in red if president sex == female*. Rezultatai matomi 41 paveikslėlyje.



41 pav Valstybės kurias valdo moterys rodomos raudonai

Siekdami panaudoti aprašomuosius duomenis apie tai, kada kurioje valstybėje paskutinį kartą buvo išrinktas prezidentas, valstybes, kuriose rinkimai įvyko nevėliau kaip

2008-11-19 nuspalvinkime žaliai. Tuo tikslu parašome veiklos taisyklę: *countries must be displayed on the map in green if date last elected > 2008-11-19*.



42 pav Valstybės kuriose vyko rinkimai atitinkame laikotarpyje

Atlikę testavimą matome, kaip paprastai neturint jokių žemėlapių valdymo programavimo žinių galima dinamiškai generuoti žemėlapius pagal kiekvienos įmonės specifinius poreikius. Žemėlapiuose gaunami rezultatai atitinka įmonių lūkesčius ir jiems pasiekti atliekami minimalūs konfigūravimo darbai, nesileidžiant iki programavimo lygio, tokiu būdu yra minimizuojamas įmonės investicijų į skaitmeninius žemėlapius kiekis.

6.5. VEIKLOS TAISYKLIŲ TIKRINIMAS

Kiekviena vartotojo įvesta veiklos taisyklė yra tikrinama ieškant įvedimo klaidų. Kompiliuojamos tik tos veiklos taisyklės kurios yra įvestos teisingai. Veiklos taisyklė yra teisinga jei tenkina šiuos kriterijus:

- Turi privalomą dalį „must be displayed “
- Nurodyta kur reikia rodyti objektus. (mūsų atveju žemėlapyje)
- Nurodyta erdvinis objektas kurį reikia vaizduoti
- Nurodytas arba nenurodytas „not“ teiginys
- Nurodyti aprašomieji duomenys
- Nurodytas objektas egzistuoja metaduomenyse
- Teisingai nurodytas vaizdavimo būdas
- Teisingai nurodyta matematinė sąlygos išraiška

Jeigu vartotojas rašydamas veiklos taisykles padaro klaidą kurioje nors iš šių dalių, duomenų integravimo komponentas nustato kurioje vietoje padaryta klaida ir praneša apie klaidą. Pavyzdžiui, panagrinėkim veiklos taisyklę „*countries must be displayed on the map in black*“. Padarykime klaidą pagrindinėje „must be displayed“ dalyje ir pažiūrėkime komponento atsaką į šią klaidą.

Klaida: Klaida pagrindinėje „must be displayed“ dalyje.

Klaidinga taisyklė: „*countries must be dispyed on the map in black*“

Komponento atsakymas: „*business rule does not have the main "be displayed" part. Rule='countries must be dispyed on the map in black'*“ Matome, kad komponentas parodo veiklos taisyklę kurioje aptikta klaida ir vietą kurioje klaida yra.

Žemiau pateiktoje lentelėje 2 yra surašytos galimos vartotojo įvedimo klaidos ir komponento reakcija į jas.

2 lentelė. Komponento atsakas į galimas klaidas

Gera taisyklė	Bloga taisyklė	Padaryta klaida	Komponento atsakas
countries must be displayed on the map in black	*countries must be dispyed on the map in black*	Klaida pagrindinėje „must be displayed“ dalyje	*business rule does not have the main "be displayed" part. Rule='countries must be dispyed on the map in black'*

countries must be displayed on the map in black	*countries must be displayed must be displayed on the map in black*	Klaida pagrindinėje „must be displayed“ dalyje. Nurodoma per daug must be displayed komandų	*business rule have too many main "be displayed" parts. Rule='countries must be displayed must be displayed on the map in black'*
countries must be displayed on the map in black	*countries must be must be displayed on the map in black*	nurodoma per daug veiksnių „must“	*what action is not properly defined in: 'countries must be must'*
countries must be displayed on the map in black	*countries must nott be displayed on the map in black*	blogai nurodomas „not“ neiginys	*'not' statment is not properly defined in: 'nott '*
countries must be displayed on the map in black	*must be displayed on the map in black*	nenurodomi erdviniai duomenys	spatial name not defined in: ' '
countries must be displayed on the map in black	*countries must tr be displayed on the map in black*	nereikalingi simboliai tarp veiksmo nurodymo ir pagrindinės dalies	*there must be no other words between what action and action in: ' tr '*
countries must be displayed on the map in black	*countries be displayed on the map in black*	nenurodomas veiksmas	*there is no 'what action' specified in: 'countries '*
countries must be displayed on the map in black	*countries must be displayed on the in black*	nenurodoma kur vaizduoti	*there is no display type in ' the in black '*
countries must be displayed on the map in black	*countries must be displayed sd on the map in black*	blogai nurodoma kur vaizduoti	*'where action' is not found or it's defined not properly in 'sd on the map in black'*
countries must be displayed on the	*countries must be displayed on the*	taisyklėje per daug sąlygų	*there is too many conditions in ' in green if*

map in green if date last elected > 2008-11-19	*map in green if esf if date last elected > 2008-11-19*		*esf if date last elected > 2008-11-19 '*
countries must be displayed on the map in green if date last elected > 2008-11-19	*countries must be displayed on the map in green if dgfdate last elected > 2008-11-19*	blogai nurodyti aprašomieji duomenys	*cant compile busines rule, data name or spatial name not exists in metadata properties: 'dgfdate last elected' or 'countries'*
countries must be displayed on the map in green if date last elected > 2008-11-19	*countries must be displayed on the map in greeeen if date last elected > 2008-11-19*	blogai nurodytas vaizdavimo būdas	*display manear not recognized manear='in greeeen'*
countries must be displayed on the map in green if date last elected > 2008-11-19	*countries must be displayed on the map in green if date last elected > 2008-11---19*	bloga matematinė išraiška	*mathematical expression is not found in: 'date last elected > 2008-11---19 '*

7. DARBO REZULTATŲ APIBENDRINIMAS

Atlikus dinamiškos GIS sistemos testavimą yra būtina nustatyti ar darbe užsibrėžti kriterijai yra išpildyti. Tam, kad įrodytume, jog darbo tikslai yra patenkinti ir numatyti rezultatai yra pasiekti, sudarysime lentelę, kurioje palyginsime sukurtą dinamišką GIS sistemą su analizuotomis analizės dalyje. Lentelė yra panaši į lentelę nr1. Skirtumas yra tas, kad šalia anksčiau apibendrintų GIS sistemų pridėsime šiame darbe sukurtą dinamišką GIS sistemą ir pažiūrėsime kaip ji atitinka darbo pradžioje nustatytus kriterijus. Papildomam stulpelyje yra trumpai paaiškinama, dėl kokios priežasties dinamiška GIS sistema tenkina arba netenkina vienokį ar kitokį kriterijų.

3 lentelė. Esamų ir sukurto sprendimo palyginimas

	GEOLIS	**VERSLO GIS**	**AUTODESK**	**GOOGLE MAPS**	**DINAMIŠKA GIS**	**Funkcionalumas dinamiškoje GIS**
Sprendimo naujumas	-	-	-	-	+	Sprendimas unikalus ir paremtas veiklos taisyklėmis
Pakartotinas panaudojimas	-+	-+	+	+	+	Duomenų integravimo komponentas yra sukompiliuojamas į *.jar bylą, kuri gali būti pakartotinai panaudojama vartotojų taikomosiose programose
Panaudojimo paprastumas	-+	-+	+	+	+	Duomenų integravimo komponentą reikia tik įdėti į savo taikomąją programą bei tinkamai sukonfigūravus per išorines bylas iškvieti *integrate()* metodą rezultatams gauti
Konfigūravimo paprastumas	nėra	nėra	-+	-+	+	Konfigūravimas atliekamas tiesiogiai arba iš taikomosios programos rašant duomenis į rules.prperties ir dataConfig.prperties bylas
Architektūros išskaidymas	+	+	+	+	+	Sistemos architektūra gali būti išskaidyta per keletą vienas nuo kito nutolusių serverių.
Esamų duomenų panaudojimas	+	+	+	+	+	Į sistemos erdvinių duomenų bazę Oracle 10g galima importuoti jau egzistuojančius erdvinius duomenis
Naujų duomenų kūrimas	+	+	+	+	+	Į sistemos erdvinių duomenų bazėje Oracle 10g galima sukurti naujus duomenis
Nepriklausomumas nuo DB	nėra	nėra	nėra	nėra	+	Verslo aprašomieji duomenys gali būti paimami iš duomenų bazės automatiškai nepriklausomai nuo duomenų bazės tipo
Nepriklausomumas nuo verslo duomenų	nėra	nėra	nėra	nėra	+	Verslo aprašomieji duomenys interpretuojami nepriklausomai nuo jų struktūros
Naujų vaizdavimo	-	-	-	-+	-+	Įtraukus naujas vaizdavimo galimybes pakaktų klientams

galimybių įtraukimas						atnaujinti API savo pusėje ir parašyti naujas veiklos taisykles
Dinamiškumas	-	-	-	-+	+	Verslo ir erdviniai duomenys interpretuojami ir integruojami automatiškai atvaizduojant juos skaitmeniniame žemėlapyje pagal sukompiliuotas veiklos taisykles
Rezultatų aiškumas	-+	+	-+	+	+	Duomenys rodomi žemėlapyje atitinkamomis spalvomis, šalia generuojama legenda.
Programavimo darbų minimizavimas	-	-	+	-+	+	Siekiant panaudoti skaitmeninius žemėlapius vartotojo taikomojoje programoje nereikia atlikti programavimo darbų, reikia tik tinkamai sukonfigūruoti
Investicijų mažinimas	-	-	-+	-+	+	Kadangi komponentas yra pakartotinai panaudojamas(be programavimo darbų) ir įmonėms nereikia jokios programinės įrangos susijusios su erdvine informacija bei skaitmeniniais žemėlapiais, todėl įmonės investicijos į tokią GIS sistemą yra minimalios

Pagal 3 lentelėje pateiktus vertinimo duomenis matome, kad sukurta dinamiška GIS sistema atitinka užsibrėžtus kriterijus ir ji gali būti pranašesnė už esamus sprendimus, jos veikimo principas yra unikalus ir dinamiškas, paremtas veiklos taisyklėmis, todėl jis nereikalauja atlikti programavimo darbų siekiat susieti erdvinius ir aprašomuosius duomenis žemėlapyje, bei reikalauja minimalių pastangų jo konfigūravimui verslo reikalavimams pasikeitus, todėl niekada nereikia keisti programos kodo susijusio su skaitmeninių žemėlapių generavimu.

IŠVADOS

1. Šiame darbe sprendžiama mažoms ir vidutinėms įmonėms aktuali skaitmeninių žemėlapių panaudojimo problema. Esami GIS sprendimai nesuteikia efektyvių įmonės aprašomosios informacijos vaizdavimo skaitmeniniame žemėlapyje galimybių.
2. Žemėlapių panaudojimui patobulinti sukurtas veiklos taisyklėmis pagrįstas komponentas, kuris dinamiškai integruoja įmonių aprašomuosius duomenis su erdvine informacija saugoma nutolusioje duomenų bazėje.
3. Operavimui erdvine informacija pasirinkta ir sukonfigūruota Oracle DBVS programinė įranga, kuri užtikrina greitą ir optimalų operavimą erdviniais duomenimis, bei žemėlapių generavimą.
4. Darbe aptarta esamų erdvinių duomenų kūrimo ir importavimo į duomenų bazę galimybė. Šis aspektas yra svarbus kadangi didžioji dalis erdvinės informacijos jau egzistuoja ir jos nereikia kurti, todėl pakanka surinktus didelius erdvinės informacijos kiekius automatizuotu būdu importuoti į Oracle DBVS duomenų bazę.
5. Sukurtas naujas sprendimas yra dinamiškas ir universalus, pritaikytas bet kuriai verslo krypčiai ir gali būti pakartotinai panaudojamas. Erdviniai ir verslo duomenys yra integruojami dinamiškai, todėl norit sugeneruoti skaitmeninį žemėlapį nereikia papildomų žmogiškųjų resursų.
6. Veiklos taisyklių panaudojimas supaprastina duomenų integravimo komponento konfigūravimą. Todėl komponentu naudotis gali informacinių sistemų kūrėjai neturintys papildomų žinių apie darbą su erdvine informacija bei žemėlapių generavimu.
7. Pasiūlytas sprendimas gali būti pritaikytas darbui su kitomis DBVS, kurios paliko OpenGIS standarto nustatytus duomenų tipus skirtus erdvinių duomenų saugojimui .

TERMINŲ ŽODYNĖLIS

GIS - Sistema, kuri skirta darbui su erdvine ir aprašomąja informacija.

API - Programavimo sąsaja yra komplektas komandų, funkcijų, ir protokolų, kuriuos programuotojai gali panaudoti, kurdami programinę įrangą priėjimui prie komponento kuriam priklauso šis API

DBVS – Duomenų bazių valdymo sistema

SQL - yra kalba, kuri suteikia sąsają su reliacinėmis duomenų bazių sistemomis

PL/SQL – Oracle SQL plėtinys

LBS - Vieta pagrįstos paslaugos yra paslaugos, kurios eksploatuoja žinias apie tai, kurioje vietovėje informacijos prietaiso vartotojas yra.

JDBC - Java Duomenų bazės Sujungimo galimybė yra programavimo sąsaja taikomosioms programoms sujungti su duomenų bazėmis.

ODBC yra standartizuota taikomosios programinės įrangos(dalykinių programų) programavimo sąsaja (API) prisijungimui prie duomenų bazių (RDBMS).

OracleAS - Taikomasis Oracle Serveris yra taikomųjų programų platformos rinkinys, kuris siūlo pilną J2EE palaikymą.

OLE DB - programavimo sąsaja, suprojektuota Microsoft tam, kad gauti prieigą prie skirtingų duomenų tipų.

AJAX - terminas, apibrėžiantis svetainių programavimo technologiją, naudojančią Asinchroninę JavaScript ir XML priemones maksimaliam interaktyvumui pasiekti.

SANTRAUKA ANGLŲ KALBA

DYNAMIC LINKING OF GEOGRAPHICAL AND ATTRIBUTIVE DATA USING ORACLE DBMS

Summary

This work reviews advantages and weaknesses of existing GIS solutions, explores usage of digital maps for rendering specific enterprise data in digital maps. Oracle DBMS software was also analyzed to prove that it has all necessary components for spatial data manipulation. After summarizing analysis results we offer new GIS solution, which allows rendering enterprise data on the map in dynamic way without knowing enterprise data structure and minimizing programmer work. To implement a dynamic GIS conception a new dynamic GIS architecture and the missing component design for representing business data in a map are created. To ensure dynamic integration and simple usage of component, dynamic integration component functionality is based on business rules conception. To demonstrate dynamic enterprise and spatial data integration on the map, dynamic GIS architecture was implemented using Oracle software by configuring data integration component to use Oracle mapViewer. For dynamic GIS testing, test interface was created with an ability to write business rules directly in web page and see the integration results. Existing spatial data was imported into Oracle DB and some spatial data for testing purposes was created manually.

LITERATŪRA

1. A. Ionita; M. Foca; M.I. Popovici . GIS in Business Processes, 2006 [žiūrėta 2007-10-04] Prieiga per Internetą: <http://www.gisdevelopment.net/application/business/business_2.htm>
2. D. Andes. What is GIS?. (Informational site on GIS), 2007 [žiūrėta 2007-10-4]. Prieiga per Internetą: <http://www.gislounge.com>
3. Gordon N. Keating; Paul M. Rich; Marc S. Witkowski. Challenges for Enterprise GIS . LA-UR-02-1830 [žiūrėta 2007-10-4]. Prieiga per internetą: <http://gislab.lanl.gov/docs/Keating_LAUR-02-1830.pdf>
4. B. Kropla.Beginning MapServer, Open Source GIS Development.
5. Steve Walden , Sagent The Need for Operational Business Geographic Solutions, 2003 [žiūrėta 2007-12-20]. Prieiga per internetą <http://www.directionsmag.com/article.php?article_id=294&trv=1>
6. Y. V. Natis; M. Pezzini; K. Iijima, M. Barnes. Magic Quadrant for Enterprise Application Servers, 2Q06. Research G00141872, 2006 [žiūrėta 2007-10-4]. Prieiga per Internetą: <http://mediaproducts.gartner.com/reprints/oracle/141872.html>
7. (2006-09-11). Rinkos analitikai „Oracle“serverį skelbia geriausiu. IT naujienos Oracle [žiūrėta 2007-10-04]. Prieiga per Internetą: <http://www.elektronika.lt/news/computers/5391/>
8. D. Sonnen; H. D. Morris. Oracle 10 *g*: Spatial Capabilities for Enterprise Solutions. White paper, Sponsored by: Oracle Corporation, 2005.
9. J.Farley. Oracle 10g. A Location-enabled Platform for Enterprise GIS and Core Business Applications. Earth observation magazine, 2004.
10. Oracle. Oracle Spatial user guide. User’s Guide and Reference. 10g Release 2 (10.2). B14255-01, 2005.
11. Oracle. Oracle® Application Server. MapViewer User’s Guide. Release 10.1.3.1. B14036-03, 2005.
12. J. Xie; J.Sharma. Managing Spatial Raster Data using GeoRaster. An Oracle technical whitepaper, 2005.
13. Internetinė nuoroda; [žiūrėta 2007-10-20] <http://www.mapit.lt/itweb/maps.nsf/0/6DA1F36ECCA35977C22570ED004D5F1E>
14. 2006 m. balandžio 4 d. "VERSLO ŽINIŲ" PASKUTINIS PUSLAPIS: Planuodami verslą kauniečiai gali pasitelkti ir geografinę informacinę sistemą [žiūrėta 2007-10-

23]. Prieiga per internetą:
<http://www.it.lt/itweb/affecto.nsf/Akt4/E41E14D6F409FECBC225714600287AEA>
15. Internetinė nuoroda [žiūrėta 2008-12-20].
<http://code.google.com/apis/maps/documentation/index.html>
16. Roland G.Ross. Principles of the Business Rule Approach. Addison-Wesley informatikon technology series, 2003 02 (123-161psl.).
17. Chris Bradshaw. Autodesk and Oracle: Complete Solutions for GIS. Infrastructure Solutions Division, 2005.
18. ESRI, ArcSDE Configuration and Tuning Guide for Oracle, 2003;
19. ESRI and Oracle — Solutions for GIS and Spatial Data Management, 2001

1 . PRIEDAS. KONFERENCIJOJE PRISTATYTAS IR IŠSPAUSDINTAS STRAIPSNIS

Dinamiškas erdvinių ir verslo duomenų integravimas panaudojant Oracle dbvs

Giedrius Racibara
Kauno technologijos universitetas
Informacijos sistemų katedra, Studentų 50-308

Straipsnyje apžvelgiama egzistuojančių GIS sistemų sprendimų privalumai ir trūkumai. Pateikiamas sprendimas kuris išspręstų dinamiško erdvinių ir verslo duomenų integravimo problemą. Pristatomos Oracle DBVS panaudojimo galimybės minėtos problemos sprendime, bei suprojektuota dinamiškos GIS sistemos veikimo schema. Tokios sistemos veikimui užtikrinti suprojektuotas komponentas, kurio veikimo principas paremtas veiklos taisyklių koncepcija. Komponento veikimui patikrinti, pateiktas teorinis tokios GIS sistemos panaudojimo pavyzdys.

Įvadas

Sėkmingam verslo funkcionavimui nepakanka vien tik sukaupti duomenis apie įmonės veiklą. Efektyvus operavimas duomenimis ir greitas bei aiškus rezultatų pateikimas yra aktuali problema šių dienų įmonėse nepriklausomai nuo veiklos srities. Nenaudojant specialių priemonių duomenų valdymas darosi sudėtingas. Erdvinės informacijos analizės naudą pripažįsta ir vis labiau vertina verslo įmonės ir organizacijos, per GIS technologijas atrasdamos naujus verslo sprendimus, klientus ir tolimesnės plėtros galimybes [1].

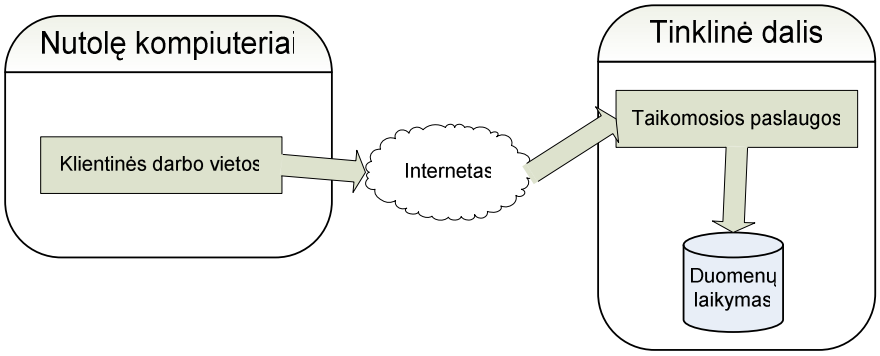
Dabartiniu metu daug įmonių turimų duomenų siejasi su geografiniais duomenimis, tačiau šis aspektas neišnaudojamas pagal galimybes ir galimą paskirtį. Tokie duomenys gali būti kliento adresai, komercinės teritorijos ir t.t. Įmonės dažnai turi spręsti klausimus, pagrįstus vietovės ryšiais tokiais kaip pardavimų kiekis konkrečioje teritorijoje, aptarnaujančio personalo vieta artima aptarnavimo taškui, ar netgi pirmo atsiliepiančiojo į SOS signalą radimas [2].

1. GIS sistemos sandara

Geografinė informacinė sistema (GIS) – informacinė sistema skirta darbui su erdvine ir aprašomąja informacija. Sistema skirta skaitmeninių koordinuotų erdvėje duomenų kaupimui, saugojimui, vaizdavimui, redagavimui, integravimui bei analizei.
GIS sudaro programinė įranga, kuri jungia geografinę informaciją (kur objektai lokalizuoti) su aprašomąja informacija (į ką objektai panašūs). Skirtingai nuo plokščio popierinio žemėlapio, GIS gali turėti daug sluoksnių informacijos po jo paviršiumi. Tokiu būdu galima matyti struktūrą ar tendencijas verslui būdingoje informacijoje, rodyti ir apskaičiuoti optimalius transporto priemonės maršrutus, sukurti norimas linijas ir daug daugiau.

Tipinę GIS sistemos architektūrą sudaro trys skirtingos dalys [3]:

- Duomenų laikymas – apima įvairių erdvinių ir aprašomųjų duomenų sandėliavimą organizacijos viduje. Šitie duomenų sandėliai paprastai turi savyje informaciją keliuose nesuderinamuose duomenų formatuose.
- Taikomosios paslaugos - susideda iš GIS Taikomojo Serverio ir Tinklo Serverio, kurie yra atsakingi už duomenų prieigą, apdorojimą ir vartotojo aptarnavimą per kliento darbo vietą (klientinę programinę įrangą). Tinklo Serveris, tai GIS Tinklo Serveris, kuris leidžia komunikuoti su vartotojų standartiniais interneto serveriais tokiais kaip Apache, Microsoft IIS ir pan. Taikomasis GIS Serveris yra funkcinis GIS paslaugų pagrindas jis suteikia erdvinių duomenų integravimą ir transformavimą.
- Klientinė darbo vieta – klientais gali būti visi informacinių sistemų vartotojai esantys bet kuriame pasaulio taške. Klientas paprastai kreipiasi į GIS taikomąsias paslaugas prašydamas atvaizduoti tam tikrą erdvinių duomenų sluoksnį ir jame esančius objektus. Kliento darbo vieta gali buti interneto naršyklės, mobilieji įrenginiai ir t.t.



1. pav. GIS sistemos architektūra

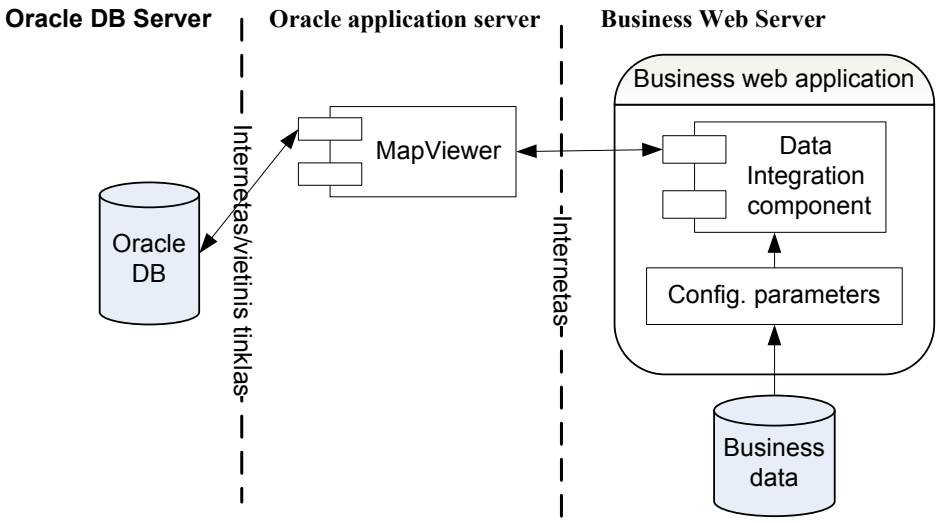
Ši trijų dalių architektūra yra ideali įmonės sistemoms, nes ji įgalina portatyvių ir išplečiamų sistemų konstravimą. Ji leidžia GIS sistemoms supaprastinti skirtingų duomenų šaltinių integraciją, suteikiant funkcionalumo karkasą, kuris leidžia keistis žemėlapiais ir duomenimis tinkle.

2. Dinamiškos GIS sistemos koncepcija

Per paskutinius dešimtį metų sukurta daugybė GIS taikomųjų sistemų , kurios užtikrino sėkmingą erdvinės informacijos produktų pateikimą tiek viešajame, tiek ir privačiuose sektoriuose. Neabejotinai šios sistemos buvo naudingos, tačiau dauguma organizacijų vis dar susiduria su GIS aplikacijų diegimo savo veiklos srityje problemomis. Pavyzdžiui, Kauno miesto teritoriją apimantis Verslo GIS sprendimas [4] vartotojui suteikia daug informacijos apie Kaune esančias įmones. Jų pateikiamą informaciją galima peržiūrėti įvairiais pjūviais, tačiau naudojimasis šia informacija tampa ribotas, kadangi ją galime tik peržiūrėti, o rezultatus turime atsirinkti patys ir negalime patys pasirinkti rezultatų vaizdavimo būdo. Panašaus tipo sistemų apstu tiek Lietuvoje, tiek ir užsienyje (Maps.lt Žemėlapis.lt, multiMap ir t.t.). Google maps API suteikia lanksčias funkcijas, kurių dėka galima operuoti Google duomenų bazėje sukaupta erdvine informacija. Tačiau šių funkcijų naudojimas reikalauja papildomų programavimo žinių ir darbų norint pridėti naują funkciją konkrečiame vartotojo internetiniame puslapyje kuriame jis pateikia erdvinę informaciją.

Mažos ir vidutinio dydžio įmonės nori naudotis optimaliais GIS sprendimais už mažiausią kainą. Dėl šios priežasties kiekviena įmonė negali sau leisti įsigyti nuosavų skaitmeninių žemėlapių bei reikiamos programinės įrangos jų administravimui bei apdorojimui. Akivaizdu, kad yra tikslinga sukurti vieną erdvinės informacijos saugyklą, kurioje būtų talpinama erdvinė informacija ir šia informaciją galėtų naudoti daugelis įmonių. Taip pat yra būtina maksimaliai sumažinti programinės įrangos kiekį, kurį turi įsidiegti įmonė tam, kad savo taikomosiose programose galėtų naudotis GIS paslaugomis. Norint įgyvendinti šiuos tikslus, reikia sukurti internetinę paslaugą, kuri sugebės aptarnauti įmonių poreikius nuotoliniu būdu.

Siekiamas sprendimas realizuojamas naudojantis Oracle programine įranga. Ši įranga papildoma nauju komponentu, kuris realizuoja duomenų integravimo dalį. Kuriamos GIS sistemos sudedamosios dalys buvo paminėtos ankstesniame skyrelyje, 2 paveikslėlyje pateikiama schema, kuri parodo kaip šios dalys susiję tarpusavyje.



2. pav Dinamiškos GIS sistemos architektūra

Oracle DataBase – Oracle DBVS serveris, kuriame saugomos duomenų bazės su erdvine informacija [5].

Oracle application server – taikomųjų uždavinių serveris, kuriame veikia MapViewer komponentas, turintis galimybę komunikuoti su **Oracle DataBase** serveriu ir naudoti erdvinius duomenis žemėlapių generavimui. [6]

Business Web Server – šiame serveryje yra vartotojo taikomojo programa, kuri naudoja duomenų integravimo komponentą. Vartotojo taikomoji programa aprūpina komponentą reikalingais parametrais, kad šis galėtų atlikti duomenų integravimo funkciją.

3. Dinamiškos GIS sistemos modelis

Kuriama GIS sistema turės palengvinti verslo duomenų ir erdvinės informacijos integravimą nepriklausomai nuo duomenų struktūros. Tokiu būdu naudojantis ta pačia sistema ir tais pačiais erdviniais duomenimis, skirtingo tipo verslo duomenys galės būti susiejami su erdvine informacija.

Operavimui erdvinais duomenimis buvo pasirinkta Oracle DBVS, kadangi ji turi visas reikalingas funkcijas bei komponentus, kurie užtikrina erdvinės informacijos įrašymą, saugojimą, skaitymą bei apdorojimą formuojant skaitmeninius žemėlapius su Ajax funkcionalumu.

Šios paslaugos realizavimui reikalingi šie komponentai:

1. Erdvinių duomenų (Oracle spatial) serveris
2. MapViewer serveris
3. Duomenų integravimo komponentas

3.1. Erdvinių duomenų (Oracle spatial) serveris

Oracle Spatial yra Oracle DBVS komponentas, kuris suteikia galimybę naudotis efektyviais GIS ir LBS (Vietovės nustatymo paslaugos) sprendimais ir yra pagrindas sudėtingoms GIS taikomosioms programoms, kurios reikalauja didelės erdvinės informacijos apdorojimo spartos. Oracle Spatial palaiko visus pagrindinius erdvinės informacijos tipus bei duomenų modelius.

3.2. MapViewer serveris

Taikomasis Oracle Serveris MapViewer (OracleAS MapViewer) yra programuojamas įrankis kuris skirtas tam, kad generuotų žemėlapius, naudodamas erdvinius duomenis, valdomus Oracle Spatial ar Oracle Locator (taip pat vadinamo tiesiog Locator). OracleAS

MapViewer suteikia įrankius, kurie paslepia erdvinių duomenų užklausų ir žemėlapio generavimo sunkumą, suteikdami įvairių pasirinkimų galimybę labiau pažangiems vartotojams. Šitie įrankiai gali būti išdėstyti nepriklausomu nuo platformos būdu ir yra suprojektuoti taip, kad būtų integruojami su žemėlapius teikiančiomis paslaugomis.

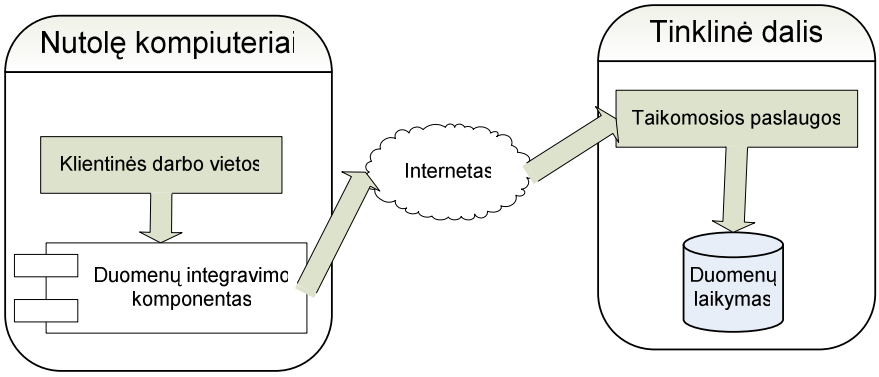
OracleAS MapViewer yra traktuojamas kaip Taikomojo Oracle Serverio dalis. Jo svarbiausia dalis yra J2EE taikomojo programa, kuri gali būti įdiegta į J2EE konteinerį.

3.3. Duomenų integravimo komponentas

Šis komponentas atlieka verslo duomenų parengimą integravimui su erdviniais duomenimis. Kadangi daugelis įmonių žemėlapius nori vaizduoti internetiniuose puslapiuose, bei suteikti tam tikrą žemėlapių ir objektų juose valdymą, komponentas suteikia sąsają, kurios dėka įmonės internetiniame puslapyje būtų galima pavaizduoti žemėlapį bei suteikti jam norimą funkcionalumą. Komponentas turi metodus, kurie susieja verslo duomenis su erdviniais duomenimis nereikalaujant didelių vartotojo pastangų. T.y. vartotojas nežino ir neturi nieko bendra su tuo, kokiu būdu ir kokiam pavidale yra saugomi erdviniai duomenys. Jis žino tik tai tą, ką jis nori matyti. Šiam tikslui pasiekti, galima iš anksto parengti veiklos taisyklių šablonus, kurių dėka verslo duomenys būtų susiejami su erdvine informacija. Veiklos taisyklių aprašymui reiktų naudoti standartizuotą metodą. Vienas iš tokių priimtiniausių metodų būtų Business Rules Solutions RuleSpeak [7]. Šis metodas turi nestruktūrinių veiklos taisyklių aprašymo šablonais galimybę. Erdvinių ir verslo aprašomųjų duomenų integravimui nestruktūrizuotų veiklos taisyklių pilnai pakanka.

3.3.1. Vieta tradicinėje GIS sistemoje

Duomenų integravimo komponentas veikia kaip tarpininkas tarp žemėlapį generuojančios programinės įrangos ir klientinės darbo vietos. Kadangi siekiama dinamiško duomenų integravimo, todėl žemėlapio generavimas taip pat turi būti dinamiškas. Tai reiškia, jog šis komponentas išanalizavęs vartotojo pateikiamus verslo duomenis ir atsižvelgęs į vartotojo nurodytus parametrus turi pateikti atitinkamą žemėlapio fragmentą. Žemėlapio vaizdavimo parametrai nurodomi veiklos taisyklių pagalba. Tokiu būdu vartotojas naudojantis šį komponentą savo aplikacijoje nieko nežino apie GIS informacijos struktūrą esančią GIS serveriuose.



3. pav. GIS sistemos architektūra

Kaip buvo pateikiama 3 pav. duomenų integravimo komponentas gali būti naudojamas kaip API įvairiose vartotojų aplikacijose. Vartotojas išlaisvinamas nuo komponento programavimo darbų. Norint tinkamai panaudoti šį komponentą tereikia jį tinkamai sukonfigūruoti:

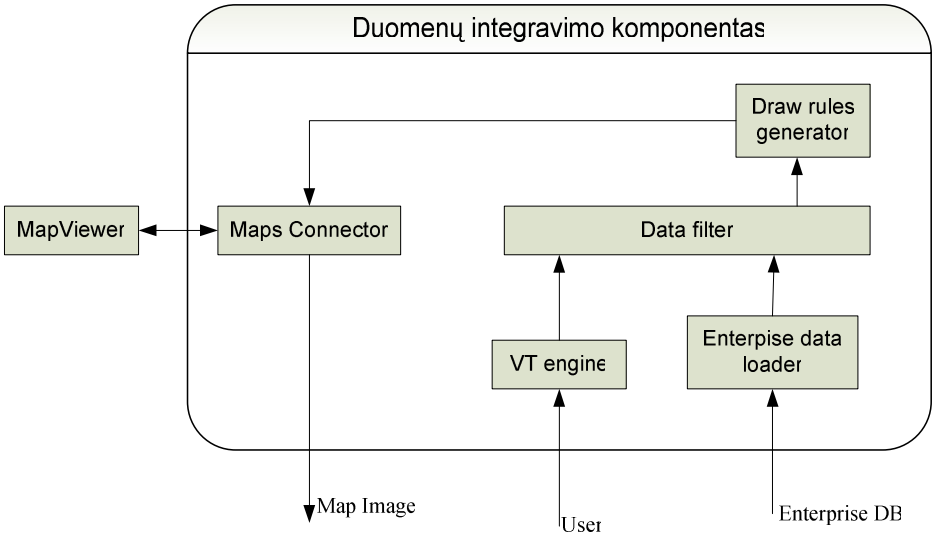
- Reikia nurodyti verslo duomenų šaltinį. Gali būti atmintyje laikomos struktūros arba informacija vartotojo duomenų bazėje.
- Reikia nurodyti prisijungimą prie žemėlapius generuojančios paslaugos. Šiuo atveju tai yra MapView‘er. Šis Oracle DBVS komponentas tradicinėse GIS sistemose yra

valdomas javaScript‘ais tokiu būdu generuojantis žemėlapius su Ajax funkcionalumu. Duomenų integravimo komponento paskirtis yra realizuoti funkcijas, kurios dinamiškai generuotų reikiamus parametrus šioms javaScript funkcijoms, tokiu būdu galima gauti dinamiškus žemėlapius.

- Reikia nurodyti taisyklių rinkinį pagal kurį būtų filtruojami verslo duomenys ir susiejami su erdviniais duomenimis pagal veiklos taisyklėse nurodytus kriterijus.

3.3.2. Veikimo principas ir architektūra

Kaip matome 4 paveikslėlyje, duomenų integravimo komponentas turi būti aprūpintas duomenų srautu iš išorinės duomenų bazės. Taip pat vartotojas turi nurodyti pagal kokias veiklos taisykles jis nori interpretuoti duomenis. Surinkus visą reikimą informaciją, įvyksta komunikacija su MapViewer‘iu ir gaunamas norimas žemėlapis.



4. pav Duomenų integravimo komponento architektūra

Vartotojo taikomojoje programoje tereikia vieną kartą nurodyti veiklos taisykles pagal kurias turės būti interpretuoti duomenys. Kitaip tariant komponentas yra įprogramuojamas vartotojo internetiniame puslapyje ir kiekviena kartą veiklos taisyklių nurodinėti nereikia. Kuriant kita puslapį galima nurodyti kitokias taisykles. Taip pat pakanka vieną karta nurodyti verslo duomenų srautą ENTERPRISE DATA LOADER moduliui. Vėliau šie duomenys bus dinamiškai integruojami su erdvine informacija pagal užduotas veiklos taisykles.

Supaprastintas veiklos taisyklių šablonas galėtų atrodyti taip: [1. Verslo/erdviniai duomenys] [2.duomenų filtras][3. ką su jais daryti].

1 – tai duomenys apibūdinantys verslo objektą kuris yra ir erdvinis objektas.
2 – kriterijus pagal kurį atrenkamas verslo duomenų objektas
3 – veiksmas ką su apibūdintais objektais daryti

Panagrinėkim kaip komponentas reaguos į veiklos taisyklę parengtą pagal pateiktą šabloną. „[1. Kelius] [2.kuriuose avarijų skaičius > 10] [3. nuspalvinti raudonai]“

1. ENTERPRISE DATA LOADER užkrauna visus komponentui skiriamus verslo duomenis.
2. VT ENGINE pagal šabloną atskiria kad iš verslo duomenų nagrinėsim tik tuos, kurie susiję su keliais ir sukuria filtrą pagal nurodyta VT.
3. DATA FILTER filtruoja verslo duomenis pagal nurodytą filtrą išrinkdamas tik tuos kelius, kuriuose avarijų skaičius didesnis nei 10.

4. DRAW RULES GENERATOR sugeneruoja kreipinį į MapViewer kuris sugeneruoja žemėlapį iš nurodytų kelių nuspalvindamas juos raudonai.

Išvados

- Google pateikia lankstų, visiems interneto naudotojams prieinamą GIS duomenų (žemėlapių) naudojimo sprendimą, tačiau jis yra ribotas. Google laisvam naudojimui atiduoda ištisus terabaitus sukauptos erdvinės informacijos, tačiau apribojimai atsiranda, kai trečioji šalis turi pati suprogramuoti kodo dalį kuri realizuoja ką, kur ir kokioje formoje norima matyti. T.y. verslo ir erdvinių duomenų integravimas yra statiškas.
- Norint operuoti erdviniais duomenimis, reikia įsidiegti visą eilę brangios programinės įrangos. Tai yra nepriimtina mažoms ir vidutinėms verslo įmonėms. Yra tikslinga šią programinę įrangą įdiegti serverio dalyje, tokiu būdu įmonės išlaisvinamos nuo papildomų investicijų. Egzistuojantys žemėlapių generavimo komponentai yra efektyvūs, tačiau pertekliniai ir per brangūs siekiant optimalaus sprendimo verslui. Žemėlapio įkėlimui į internetinį puslapį reikia naudoti kiek įmanoma paprastesnį komponentą.
- Maksimalų panaudojimo efektyvumą atitiktų toks komponentas, kurio konfigūravimui siekiant gauti norimus rezultatus reikėtų kiek galima mažiau pastangų.
- Toliau numatoma kurti ir realizuoti verslo ir erdvinių duomenų integravimo pagal veiklos taisykles algoritmą.

Literatūros sąrašas

[1] D. Andes (2007). What is GIS?. (Informational site on GIS).

[2] A. Ionita, M. Foca, M.I. Popovici (2006). GIS in Business Processes.

[3] Internetinė nuoroda. http://www.hitachisoft-gis.com/content-en/GWSENProd_Any_Arch.htm [žiūrėta 2008-04-02]

[4] 2006 m. balandžio 4 d. "VERSLO ŽINIŲ" PASKUTINIS PUSLAPIS: Planuodami verslą kauniečiai gali pasitelkti ir geografinę informacinę sistemą. Prieiga internetu: http://www.affecto.lt/itweb/affecto.nsf/Akt4/E41E14D6F409FECBC225714600287AEA [žiūrėta 2007-10-23]

[5] J.Farley. (2004). Oracle 10g. A Location-enabled Platform for Enterprise GIS and Core Business Applications. Earth observation magazine.

[6] Oracle (2005). Oracle® Application Server. MapViewer User's Guide. Release 10.1.3.1. B14036-03.

[7] Kapočius K., Butleris R. Repository for Business Rules Based IS requirements. Informatica, Vol. 17, No.4, p. 503-518., 2006

2. PRIEDAS. DUOMENŲ APRIBOJIMAI IR METADUOMENYS

Matematinės išraiškos veiklos taisyklėse atpažįstamos pagal šias reguliarias išraiškas:

Realūs ir sveikieji skaičiai(naudojant operandus <;>;<=;>=;=):

```
N_D_L_V_PATTERN = "(\\w+[ ]*)+<[ ]*-?[\\d]+.?[\\d]*[ ]*$";
N_D_LE_V_PATTERN = "(\\w+[ ]*)+<=[ ]*-?[\\d]+.?[\\d]*[ ]*$";
N_D_G_V_PATTERN = "(\\w+[ ]*)+>[ ]*-?[\\d]+.?[\\d]*[ ]*$";
N_D_GE_V_PATTERN = "(\\w+[ ]*)+>=[ ]*-?[\\d]+.?[\\d]*[ ]*$";
N_D_E_V_PATTERN = "(\\w+[ ]*)+=[ ]*-?[\\d]+.?[\\d]*[ ]*$";
N_V_L_D_L_V_PATTERN = "-?[\\d]+.?[\\d]*[ ]*<([ ]*\\w+[ ]*)+<[ ]*-
?[\\d]+.?[\\d]*[ ]*$";
N_V_LE_D_L_V_PATTERN = "-?[\\d]+.?[\\d]*[ ]*<=([ ]*\\w+[ ]*)+<[ ]*-
?[\\d]+.?[\\d]*[ ]*$";
N_V_L_D_LE_V_PATTERN = "-?[\\d]+.?[\\d]*[ ]*<([ ]*\\w+[ ]*)+<=[ ]*-
?[\\d]+.?[\\d]*[ ]*$";
N_V_G_D_G_V_PATTERN = "-?[\\d]+.?[\\d]*[ ]*>([ ]*\\w+[ ]*)+>[ ]*-
?[\\d]+.?[\\d]*[ ]*$";
N_V_GE_D_G_V_PATTERN = "-?[\\d]+.?[\\d]*[ ]*>=([ ]*\\w+[ ]*)+>[ ]*-
?[\\d]+.?[\\d]*[ ]*$";
N_V_G_D_GE_V_PATTERN = "-?[\\d]+.?[\\d]*[ ]*>([ ]*\\w+[ ]*)+>=[ ]*-
?[\\d]+.?[\\d]*[ ]*$";
```

Tekstinė eilutė (naudojant operandą =):

```
S_D_E_V_PATTERN = "(\\w+[ ]*)+==[ ]*(\\w+[ ]*)+$";
```

Data (naudojant operandus <;>;=)

```
D_D_L_V_PATTERN = "(\\w+[ ]*)+<[ ]*[\\d][\\d][\\d][\\d]-[\\d][\\d]-
[\\d][\\d][ ]*$";
D_D_E_V_PATTERN = "(\\w+[ ]*)+=[ ]*[\\d][\\d][\\d][\\d]-[\\d][\\d]-
[\\d][\\d][ ]*$";
D_D_G_V_PATTERN = "(\\w+[ ]*)+>[ ]*[\\d][\\d][\\d][\\d]-[\\d][\\d]-
[\\d][\\d][ ]*$";
```

Laikas (naudojant operandus <;>;=)

```
T_D_L_V_PATTERN = "(\\w+[ ]*)+<[ ]*[\\d][\\d]:[\\d][\\d]:[\\d][\\d][ ]*$";
T_D_E_V_PATTERN = "(\\w+[ ]*)+=[ ]*[\\d][\\d]:[\\d][\\d]:[\\d][\\d][ ]*$";
T_D_G_V_PATTERN = "(\\w+[ ]*)+>[ ]*[\\d][\\d]:[\\d][\\d]:[\\d][\\d][ ]*$";
```

Data ir laikas (naudojant operandus <;>;=)

```
DT_D_L_V_PATTERN = "(\\w+[ ]*)+<[ ]*[\\d][\\d][\\d][\\d]-[\\d][\\d]-
[\\d][\\d] [\\d][\\d]:[\\d][\\d]:[\\d][\\d].[\\d][ ]*$";
DT_D_E_V_PATTERN = "(\\w+[ ]*)+=[ ]*[\\d][\\d][\\d][\\d]-[\\d][\\d]-
[\\d][\\d] [\\d][\\d]:[\\d][\\d]:[\\d][\\d].[\\d][ ]*$";
DT_D_G_V_PATTERN = "(\\w+[ ]*)+>[ ]*[\\d][\\d][\\d][\\d]-[\\d][\\d]-
[\\d][\\d] [\\d][\\d]:[\\d][\\d]:[\\d][\\d].[\\d][ ]*$";
```

Galimi vardai duomenų bazės tipo nurodymui:
MySql, MsSql, Oracle, PostgreSql

Galimi vardai erdvinių metaduomenų nurodymui:
Roads - keliai
plc_of_int – lankytinos vietos
houses - namai
continents - žemynai
countries - šalys

Galima naudoti šias vaizdavimo spalvų išraiškas

IN BLACK, *IN CYAN*, *IN RED*, *IN WHITE*, *IN BLUE*, *IN GRAY*, *IN GREEN*, *IN MAGNETA*, *IN ORANGE*, *IN PINK*, *IN YELLOW*.

3. PRIEDAS. ERDVINIŲ OBJEKTŲ RANKINIS KŪRIMAS

Kelių kūrimas

```
CREATE TABLE roads (
  id NUMBER PRIMARY KEY,
  name VARCHAR2(32),
  Geometry SDO_GEOMETRY);
INSERT INTO roads VALUES  1,  'Savanoriu prospektas',
SDO_GEOMETRY
       (2002, 32774, -- SDO_SRID NULL, SDO_ELEM_INFO_ARRAY (1, 2, 1 ),
              SDO_ORDINATE_ARRAY (1,10, 9,10, 10,12,12,13, 14,12, 15,10, 14,8, 12,7, 12, 0)
       )
);
INSERT INTO roads VALUES(
  2, 'Tvirtovės al.',
SDO_GEOMETRY
       (2002, 32774, NULL, SDO_ELEM_INFO_ARRAY (1,2,1 ),SDO_ORDINATE_ARRAY (12,20, 12,13))
);
INSERT INTO roads VALUES(3, 'Taikos pr.',
SDO_GEOMETRY
       (2002, 32774, NULL, SDO_ELEM_INFO_ARRAY (1,2,1 ),
              SDO_ORDINATE_ARRAY (24,10, 15,10)
       )
);
```

Strateginių objektų kūrimas

```
CREATE TABLE places_of_interests (
  id NUMBER PRIMARY KEY,
  name VARCHAR2(32),
  Geometry SDO_GEOMETRY);
INSERT INTO places_of_interests VALUES(1, 'Stadionas',
SDO_GEOMETRY
       (2003, 32774,null, SDO_ELEM_INFO_ARRAY (1, 1003, 3),SDO_ORDINATE_ARRAY (2,2, 7,8 )
));
INSERT INTO places_of_interests VALUES(  2, 'Nekropolis',
  SDO_GEOMETRY(  2003,   32774,  NULL,  SDO_ELEM_INFO_ARRAY(1,1003,4),
SDO_ORDINATE_ARRAY(10,10, 12,12, 14,10)  ));
INSERT INTO places_of_interests VALUES(  3,  'Ramybes parkas',
SDO_GEOMETRY(
2003, 32774, NULL, SDO_ELEM_INFO_ARRAY -- SDO_ELEM_INFO attribute (see Table 4-2 for values)
(1, 1003, 1),SDO_ORDINATE_ARRAY
                     (7,11, 10,13, 10,17, 8,19, 5,18, 3,19, 2,17, 3,12, 7,11 )));
```

Namų kūrimas

```
CREATE TABLE houses (
  id NUMBER PRIMARY KEY,
  name VARCHAR2(32),
  Geometry SDO_GEOMETRY);
INSERT INTO houses VALUES( 1, '1',SDO_GEOMETRY
       (2003, 32774,null, SDO_ELEM_INFO_ARRAY (1, 1003, 3 ),SDO_ORDINATE_ARRAY(0,11, 2,12 )
));
INSERT INTO houses VALUES( 2,  '2',
SDO_GEOMETRY(2003, 32774,null, SDO_ELEM_INFO_ARRAY (1, 1003, 3 ),
SDO_ORDINATE_ARRAY(8,8, 10,9 )));
```

```
INSERT INTO houses VALUES(  3,  '3',
SDO_GEOMETRY(2003, 32774,null, SDO_ELEM_INFO_ARRAY (1, 1003, 3 ),
SDO_ORDINATE_ARRAY(10,5, 11,7 )))
INSERT INTO houses VALUES( 4,  '4',SDO_GEOMETRY(2003, 32774,null,
SDO_ELEM_INFO_ARRAY (1, 1003, 3 ),SDO_ORDINATE_ARRAY(10,2, 11,4 )
));
INSERT INTO houses VALUES( 5,'1',SDO_GEOMETRY(2003, 32774,null,
SDO_ELEM_INFO_ARRAY (1, 1003, 3 ),SDO_ORDINATE_ARRAY(13,16, 14,18 )
));
INSERT INTO houses VALUES(6,  '2',SDO_GEOMETRY(2003, 32774,null,
SDO_ELEM_INFO_ARRAY (1, 1003, 3 ),SDO_ORDINATE_ARRAY(13,13, 14,15 )
))
INSERT INTO houses VALUES( 7, '5',SDO_GEOMETRY(2003, 32774,null,
SDO_ELEM_INFO_ARRAY (1, 1003, 3 ),SDO_ORDINATE_ARRAY(13,5, 14,7 )));
INSERT INTO houses VALUES(  8, '6',SDO_GEOMETRY(2003, 32774,null,
SDO_ELEM_INFO_ARRAY (1, 1003, 3 ),SDO_ORDINATE_ARRAY(13,4, 14,2 )));
INSERT INTO houses VALUES(  9, '1',SDO_GEOMETRY(2003, 32774,null,
SDO_ELEM_INFO_ARRAY (1, 1003, 3 ),SDO_ORDINATE_ARRAY(16,8, 18,9 )));
INSERT INTO houses VALUES(10, '3',SDO_GEOMETRY(2003, 32774,null,
SDO_ELEM_INFO_ARRAY (1, 1003, 3 ),
SDO_ORDINATE_ARRAY(16,6, 18,7 )));
INSERT INTO houses VALUES(11, '2',SDO_GEOMETRY(2003, 32774,null, SDO_ELEM_INFO_ARRAY (1, 1003,
3 ),SDO_ORDINATE_ARRAY(19,8, 12,9 )));

INSERT INTO houses VALUES( 12,  '4',SDO_GEOMETRY(2003, 32774,null,
SDO_ELEM_INFO_ARRAY (1, 1003, 3 ),SDO_ORDINATE_ARRAY(21,11, 23,12 )));
```

4 PRIEDAS. TESTINIAI DATACONFIG.PROPERTIES BYLOS DUOMENYS

```
setDB_7=continents1
setDB_10=continents2
setDB_11=continents3
setDB_12=continents4
setDB_8=countries1
setDB_9=countries2
setDB_13=countries3
setDB_14=countries4
setDB_15=countries5

continents1.connect=jdbc:mysql://localhost:3306/continents_data
continents1.table=continents_data_sumary
continents1.spatField=cname
continents1.spatName=continents
continents1.dataField=number_of_countries
continents1.dataName=countries numb
continents1.user=root
continents1.password=Gerietis
continents1.spatMeta=continents
continents1.dbType=mySql

continents2.connect=jdbc:mysql://localhost:3306/continents_data
continents2.table=continents_data_sumary
continents2.spatField=cname
continents2.spatName=continents
continents2.dataField=number_of_people
continents2.dataName=population
continents2.user=root
continents2.password=Gerietis
continents2.spatMeta=continents
continents2.dbType=mySql

continents3.connect=jdbc:mysql://localhost:3306/continents_data
continents3.table=continents_data_sumary
continents3.spatField=cname
continents3.spatName=continents
continents3.dataField=number_of_reliies
continents3.dataName=religies
continents3.user=root
continents3.password=Gerietis
continents3.spatMeta=continents
continents3.dbType=mySql

continents4.connect=jdbc:mysql://localhost:3306/continents_data
continents4.table=continents_data_sumary
continents4.spatField=cname
continents4.spatName=continents
continents4.dataField=time_zone
continents4.dataName=time zone
continents4.user=root
continents4.password=Gerietis
continents4.spatMeta=continents
continents4.dbType=mySql

countries1.connect=jdbc:mysql://localhost:3306/continents_data
countries1.table=countries_presidents
countries1.spatField=cname
countries1.spatName=countries
countries1.dataField=sex
countries1.dataName=president sex
```

```
countries1.user=root
countries1.password=Gerietis
countries1.spatMeta=countries
countries1.dbType=mySql

countries5.connect=jdbc:mysql://localhost:3306/continents_data
countries5.table=countries_presidents
countries5.spatField=cname
countries5.spatName=countries
countries5.dataField=last_elected
countries5.dataName=date last elected
countries5.user=root
countries5.password=Gerietis
countries5.spatMeta=countries
countries5.dbType=mySql


countries2.connect=jdbc:oracle:thin:@//88.222.36.167:1521/orcl
countries2.table=LEVEL2RELIGIONS
countries2.spatField=cname
countries2.spatName=countries
countries2.dataField=rname
countries2.dataName=religion name
countries2.user=MDSYS
countries2.password=Gerietis
countries2.spatMeta=countries
countries2.dbType=oracle

countries3.connect=jdbc:oracle:thin:@//88.222.36.167:1521/orcl
countries3.table=LEVEL2RELIGIONS
countries3.spatField=cname
countries3.spatName=countries
countries3.dataField=PEOPLE
countries3.dataName=people
countries3.user=MDSYS
countries3.password=Gerietis
countries3.spatMeta=countries
countries3.dbType=oracle

countries4.connect=jdbc:oracle:thin:@//88.222.36.167:1521/orcl
countries4.table=LEVEL2RELIGIONS
countries4.spatField=cname
countries4.spatName=countries
countries4.dataField=PERCENT
countries4.dataName=percent of believers
countries4.user=MDSYS
countries4.password=Gerietis
countries4.spatMeta=countries
countries4.dbType=oracle
```