BENDRASIS PROGRAMAVIMO DOKUMENTAS
EUROPOS SĄJUNGA
SOCIALINĖS APSAUGOS IR DARBO MINISTERIJA
NACIONALINĖ ŽEMĖS TARNYBA
PRIE ŽEMĖS ŪKIO MINISTERIJOS

IŠ DALIES EUROPOS SOCIALINIO FONDO LĖŠOMIS FINANSUOJAMAS PROJEKTAS "LIETUVOS GEOGRAFINĖS INFORMACIJOS VALDYTOJŲ KVALIFIKACIJOS KĖLIMAS" (NR. BPD2004-ESF-2.2.0-02-05/0143)

Valstybės tarnautojų mokymo programa

GEOGRAFINĖS INFORMACIJOS INFRASTRUKTŪROS NUOTOLINIS MOKYMAS

Mokomoji knyga

GEOGRAFINIŲ DUOMENŲ BAZIŲ VALDYMO SISTEMOS

GII-05

Vilnius, 2008

MALASPINA UNIVERSITY-COLLEGE
VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
HNIT - BALTIC

Mokomoji knyga „Geografinių duomenų bazių valdymo sistemos“ (GII-05)

Autorius
Dave Cake

Recenzavo ir redagavo
doc. dr. Žilvinas Stankevičius (Vilniaus Gedimino technikos universitetas)

Recenzavo
doc. dr. Gintautas Mozgeris (Lietuvos Žemės ūkio universitetas)

Iš anglų kalbos vertė ir redagavo
UAB Astraneta

Valstybės tarnautojų nuotolinio mokymo programos „Geografinės informacijos infrastruktūros nuotolinis mokymas“, patvirtintos Valstybės tarnybos departamento prie LR Vidaus reikalų ministerijos direktoriaus įsakymu „Dėl valstybės tarnautojų mokymo programų tvirtinimo“ Nr. 27V-72 (2007-04-17), mokymo kursų medžiaga parengta vykdant iš dalies Europos Sąjungos lėšomis finansuojamą projektą „Lietuvos geografinės informacijos valdytojų kvalifikacijos kėlimas“ (Nr. BPD2004-ESF-2.2.0.-02-05/0143)

Projekto vykdytojas Nacionalinė žemės tarnyba prie Žemės ūkio ministerijos.

Projekto rangovas UAB HNIT-BALTIC, Vilniaus Gedimino technikos universitetas

Turinys

Įvadas

Šiame kurse apžvelgiamas geografinių arba erdvinių duomenų valdymas. Šis procesas daugiausia apima sukurtos duomenų bazės struktūras ir metodiką, bet yra temų, kurios priklauso geografinės informacijos sistemos (GIS) arba erdvinių duomenų bazių sričiai. Šią temą pradėsime nuo universalios duomenų bazių teorijos nagrinėjimo. Sužinosime bendruosius terminus, struktūras ir metodus, kurie naudojami bankininkystėje ir žemėnaudos planavime. Išnagrinėję duomenų bazių pagrindus, pereisime prie geografiniams taikymams būdingų klausimų.

Kurso struktūra keisis iš abstrakčios į konkrečią. Pradėsime nuo medžiagos, apibūdinančios duomenų bazių struktūras ir metodus, kurie naudojami projektuojant sudėtingas duomenų bazes. Projektavimo etapu pasistengsime neaptarti tų duomenų bazės sričių, kurios susijusios su naudojama technine ir programine įranga. Mums labiau rūpės naudojami duomenų elementai ir jų tarpusavio sąsaja. Tada aptarsime duomenų bazės projektavimą arba tikros duomenų bazės kūrimą naudojant konkrečią duomenų bazės pritaikymo programinę įrangą. Pabaigoje išmoksime pildyti šias duomenų struktūras ir jas prižiūrėti. Po to nagrinėsime programinės įrangos specifines dalis ir aptarsime programinės įrangos naudojimą eksploatuojant duomenis.

Didėjant šiuo metu apdorojamų erdvinių duomenų apimčiai, dėl technologijų skvarbos ir efektyvumo padidėjo erdvinių duomenų bazių ir efektyvaus erdvinių duomenų tvarkymo svarba. Įprastas GIS vadovas, ypač dirbantis nedidelėje organizacijoje be specializuoto GIS duomenų bazės vadovo, dažnai susidurs su duomenų bazės projektavimo klausimais arba turės į pagalbą kviestis šį darbą atliekančius konsultantus.

1 šio kurso dalyje aptariami teoriniai kurso pagrindai. Šioje dalyje apžvelgsime skirtingus duomenų bazės modelius ir terminus bei pradėsime kurti duomenų bazės modelius naudodami vieningos modeliavimo kalbos (UML) klasių diagramų žymėjimo sistemą. Išmoksime struktūrizuoti duomenų bazes, kad sumažintume duomenų integracijos problemų skaičių.

2 dalyje pereisime nuo abstrakčių duomenų bazės modelių kūrimo pagrindų prie konkretesnių įgyvendinimo klausimų. Čia išsiaiškinsime, kaip perkelti UML projektą į galutinės duomenų bazės lenteles, eilutes ir stulpelius. Dar pradėsime nagrinėti geoduomenų bazių kūrimo procesą – apibrėšime struktūras, duomenų įkėlimą į šias struktūras ir jų tvarkymą. Aptarsime geoduomenų bazės įrankius, kurie padės išsaugoti duomenų integralumą, įskaitant domenų naudojimą, potipius ir topologiją.

3 dalyje pagrindinis dėmesys bus skiriamas daugelio vartotojų geoduomenų bazėms. Išnagrinėsime pagrindinius kelių žmonių darbo su viena duomenų baze tuo pačiu metu principus. Praktinėje dalyje aptarsime programą *ArcSDE* (*Spatial Database Engine*) ir jos naudojimą. Įtrauksime diskusiją apie *SDE* programą vartotojo ir administratoriaus požiūriu.

4 dalyje bus aptariamos papildomos temos, susijusios su erdvinių duomenų bazių valdymu. Naudosime standartinę struktūrizuotą užklausų kalbą (SQL), kad galėtume valdyti duomenų bazes, ir aptarsime SQL plėtinius, kad galėtume vykdyti erdvines operacijas. Dar aptarsime erdvinių duomenų indeksavimą, laikinųjų duomenų valdymo būdus ir lygiagretaus ar paskirstyto duomenų bazių realizavimo parinktis.

1 Duomenų bazės pagrindai

Šis modulis yra teoriniai kurso pagrindai. 1 temoje apibūdinama pagrindinė duomenų bazių terminologija ir nagrinėjami pastarųjų metų duomenų bazių struktūros pasikeitimai. 2 temoje pagrindinis dėmesys skiriamas duomenų modelio arba duomenų bazės struktūros apibūdinimo procesui, kad būtų galima naudoti duomenis konkrečiais tikslais. Išnagrinėsime duomenų struktūrų žymėjimą ir įgausime praktinės modelių kūrimo patirties. Sukūrę modelį aptarsime dažniausiai naudojamą duomenų bazės struktūrą, reliacinį modelį. Sužinosime gero duomenų bazių projektavimo pranašumus, sumažinančius duomenų vientisumo problemas, ir išmoksime nustatyti, ar mūsų sukurto projekto struktūra yra tinkama.

Kurso dalies turinys:

- Duomenų bazės pagrindai ir terminologija
- Duomenų bazės projektavimas
- Reliacinis modelis

1.1 Pagrindai ir terminija

Pradėsime nuo bendriausių pagrindų: kas yra duomenų bazė, kuo svarbus duomenų bazės projektavimas ir susipažinsime su tam tikra terminija. Dar aptarsime kai kuriuos atskirus duomenų bazių sistemų ir duomenų modelių tipus.

1.1.1 Duomenys, informacija ir žinios

Duomenys yra faktai. Duomenų vienetas yra simbolis, naudojamas ką nors apibūdinti, pvz., žodį, skaičių ar datą. Neapdoroti duomenys yra faktai, apibūdinantys asmenis, vietas, daiktus ar įvykius. Kai kuriose nuorodose *informacija* ir *duomenys* bus vartojami sinonimiškai, bet šioje jų reikšmes išskirsime. **Informacija** – duomenys esantys prasmingame kontekste arba apdoroti ar pateikti gavėjui taip, kad turėtų prasmę.

Toliau pateiktoje 1-1 lentelėje galime aiškiai matyti **vardų** ir **skaičių** sąrašą, bet negalime suprasti jų reikšmės be pateikto konteksto.

1-1 lentelė. Duomenys be konteksto

DeNiro, R.	49333223	C
Bunyan, P.	33456327	A
Gretzky, W.	22004837	B

Tai yra faktų arba duomenų sąrašas. Šia forma ir be paaiškinimo pateikti duomenys yra beverčiai. 1-2 lentelėje tie patys duomenys pateikti su kontekstu.

1-2 lentelė. Duomenys su kontekstu. Informacija

Vardas	Studento numeris	Pažymių vidurkis
DeNiro, R.	49333223	C
Bunyan, P.	33456327	A
Gretzky, W.	22004837	B

Kai yra kontekstas, galime nesunkiai suprasti, kad tai yra studentų sąrašas ir kurso pažymių vidurkis. Ši informacija naudinga tam, kas turi priimti sprendimą dėl kursą baigusių ir nebaigusių studentų. Štai tada duomenys tampa informacija.

Tai vienas iš pavyzdžių, kai daugiareikšmis žodis vartojamas siauresne prasme. Šiuo atveju tai sąvoka *informacija*. Žmonėms, valdantiems informacijos sistemas informacija turi turėti reikšmę. GIS sistemoje dažnai susidursime su reiškiniu, kai daugiareikšmiai žodžiai, vartojant juos konkrečioje srityje, įgauna labai siaurą reikšmę. Kiekvienas mokslas turi savo kalbą, todėl GIS mokymosi dalis yra aiškus terminijos supratimas.

Žinios įgyjamos įsisavinus informaciją. Informacija ar duomenys gali būti suprantami skirtingai, atsižvelgiant į jau esamas asmens, kuris interpretuoja informaciją, žinias. Informaciją galima gauti ją išgirdus, o žinios kaupiamos mąstant. Todėl naujų žinių galima įgyti nesužinojus naujos informacijos.

Tarkime, jums pranešė, kad 2006 metų birželio 11 dieną Vilniuje buvo 10^{o}C šilumos. Informacijos kontekstas yra matų sistema ir matavimo laikas bei vieta. Tai yra informacija, nes pateikti duomenys (10) yra reikšmingi. Žinios yra papildoma reikšmė ar įžvalga, gaunama iš skaitytojo jau esamo šios informacijos suvokimo. Pavyzdžiui, jei 10 metų kiekvieną dieną jus informuotų apie Vilniuje esančią temperatūrą, jūs suprastumėte, kad 10^{o}C birželio mėnesį yra tikrai vėsoka.

Šiame kurse pagrindinį dėmesį skirsime informacijai (kaip duomenų bazėms), o ne žinioms (kaip sprendimais pagrįstose sistemose ar ekspertų sistemose).

Tinkama informacija turi šiuos 6 požymius:

Tikslumas:	be klaidų
Išsamumas:	nurodyti visi susiję faktai
Pateikimas laiku:	pasiekiama prireikus
Patikrinamumas:	galima patikrinti jos teisingumą
Ekonomiškumas:	išlaidų ir gautos naudos, duomenų surinkimo ir priežiūros išlaidų bei organizacijai teikiamos naudos balansavimas
Aktualumas:	svarbi sprendimus priimančiam asmeniui, pateikta reikiama forma (skaitine / grafine), apimanti reikiamą diapazoną, neturinti informacijos perkrovos

Paskutiniai du požymiai ypač svarbūs apdorojant duomenų bazėje saugomą skaitmeninę informaciją. Organizacija visada būna sukaupusi daugiau duomenų, negu galima saugoti duomenų bazėje. Vienas iš duomenų bazių projektuotojų uždavinių yra atskirti, kuri informacija yra svarbi, nes ji susijusi su vykdoma užduotimi.

Pavyzdžiui, koledžo registro duomenų bazėje mums reikės saugoti informaciją apie studentus, jų aktyvumą, baigtus kursus ir gautus įvertinimus. Kita studentų informacija, pvz., vairuotojo pažymėjimo numeris, darbo informacija, šeiminė padėtis, vaikų skaičius ir t. t., irgi bus kaupiama, bet svarbu nuspręsti, ar ši papildoma informacija bus naudinga registravimo valdybai. Išsaugodami visą kitą informaciją užpildysime mūsų duomenų bazę ir sulėtinsime jos veikimą, bet nepapildysime jos funkcionalumo. Be to, informacijos naujinimas ir tikslinimas yra labai brangus darbas, todėl duomenų bazėje reikia saugoti tik reikiamą informaciją.

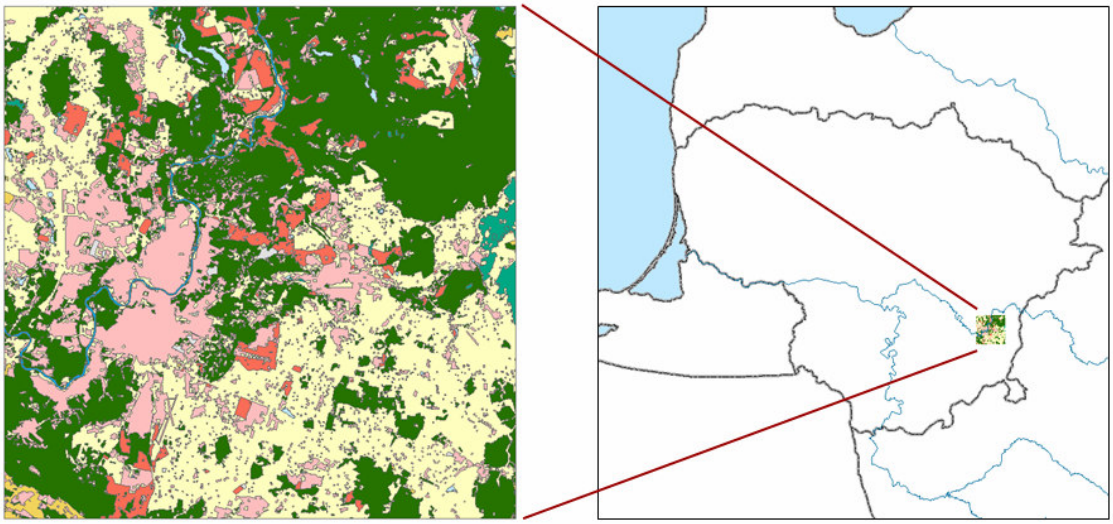
Duomenų bazė yra gerai parengtas susijusių duomenų, reikalingų įgyvendinti tam tikrą tikslą, rinkinys. Duomenys tvarkomi taip, kad būtų galima juos pasiekti ir keisti. **Duomenų bazės valdymo sistema** (DBVS) yra programinė įranga, skirta tvarkyti informaciją. DBVS leidžia **apibrėžti, sudaryti** ir **valdyti** duomenų bazę.

Duomenų bazės *apibrėžimas (defining)* yra tuščios struktūros, kurioje bus įvesti duomenys, kūrimo procesas. Nurodysime duomenų tipą (pvz., skaičius, tekstas ar data), galimų reikšmių apribojimus (pvz., amžiaus reikšmė turi būti didesnė negu 0) ir į duomenų bazę įvestos informacijos ryšius (pvz., kokiam kursui priklauso studentas). Duomenų

bazės *sudarymas* yra jos struktūrų užpildymas duomenimis. DBVS programinė įranga rašys reikšmes į savo kontroliuojamas laikmenas (pvz., kompiuterio diską) ir iš jų nuskaitys. Duomenų bazės *valdymas* yra DBVS gebėjimas gauti reikiamą duomenų bazės informaciją (**užklausa**) arba atnaujinti joje esančias reikšmes.

DBVS turi užtikrinti **efektyvų**, **patogų** ir **saugų kelių vartotojų didelio** ir **nuolatinio** duomenų kiekio saugojimą ir prieigą prie jų.

Didelis duomenų kiekis nurodo daugelio duomenų bazių dydį. Daugelis duomenų bazių, pvz., naudojamos bankininkystėje, yra kelių gigabaitų dydžio. Duomenų bazės dar padidėja, jei jose saugoma visų operacijų retrospektyva. GIS duomenų bazės yra ypatingai didelės. Pavyzdžiui, įsivaizduokite Lietuvos LTDBK50000 žemėnaudos duomenų rinkinį. Jame yra poligonų, apibūdinančių panašaus žemės naudojimo vietas, pvz., žemės ūkio paskirties vietoves, miškus ar užstatytas vietoves. Bendrąjį 1:50 000 duomenų rinkinį gali sudaryti keli tūkstančiai poligonų, kurių vieno žemėlapio lapo dydis yra keli megabaitai duomenų. Šalį sudaro 135 žemėlapio lapai, todėl visos šalies žemėnauda užima apie šimtą megabaitų. Išsamesnio atvaizdavimo failai, pvz., 1:10 000, bus daug didesni.



Elementas: vienas žemėnaudos žemėlapio lapas

Apžvalga: vienas lapas šalies kontekste

1-1 pav. Žemėnaudos duomenų bazės pavyzdys

Nuolatiniai reiškia, kad duomenys gyvuoja ilgiau negu jų valdymui naudojama programinė įranga. Dažnai programinė įranga būna populiari kelis metus, kuriems praėjus jos vieta užima kitų populiarių gamintojų programinė įranga. GIS yra geras pavyzdys, nes tam tikros GIS vartotojų grupės per 15 metų kelis kartus pakeitė naudojamas programas. Kiekvieną kartą keičiant GIS programinę įrangą, turi būti perskaičiuojami arba formatuojami iš naujo visi saugomi duomenys, kad juos būtų galima naudoti nauja GIS programa. Taip yra todėl, kad skirtingos GIS programos naudoja skirtingas duomenų struktūras. Duomenų perkėlimas iš vienos programinės įrangos programos į kitą dažnai yra ilgas ir sudėtingas procesas. Šiuo metu taip nutinka rečiau, nes yra sukurta duomenų perkėlimo tarp pagrindinių GIS programų priemonių, bet duomenų formatų keitimas visuomet pareikalaus nemažai darbo.

Keli vartotojai reiškia duomenų bazių gebėjimą palaikyti daug žmonių ar programų, turinčių prieigą prie tos pačios duomenų bazės ar net tų pačių duomenų. Kelių vartotojų duomenų bazės naudojimą būtina įdėmiai kontroliuoti. Bendras įdėmios kontrolės poreikio pavyzdys yra tuo pačiu metų atliekami nuskaičiavimai iš banko sąskaitos.

Duomenų bazė turi būti *apsaugota* nuo sistemos gedimų ir piktavalių naudotojų. Didelė dalis DBVS programų funkcijų skirta kurti atsarginių duomenų kopijas ir atkurti duomenis. DBVS ne tik tvarko efektyvias ir reguliariai kuriamas atsargines duomenų kopijas, bet ir gali valdyti operacijas. Apie tai daug išsamiau pakalbėsime vėliau, bet atliekant beveik visus duomenų veiksmus kuriami taip vadinami saugos taškai, kada išsaugomas duomenų vientisumas ir atsiradus klaidoms galima iš naujo atkurti neužbaigtas operacijas naudojant vėliausio saugos taško informaciją.

Pavyzdžiui, išimant 100 Lt iš bankomato operacija apima dvi užduotis: 100 Lt nuskaičiuojama iš jūsų sąskaitos ir jums pateikiama 100 Lt grynųjų pinigų. Jeigu vykdant šią operaciją kiltų DBVS veikimo problemų (sutriktų elektros energijos tiekimas, sugestų duomenų saugojimo laikmenos ir t. t) galimos situacijos, kad pinigai būtų nuskaičiuoti iš jūsų sąskaitos, bet jūs negautumėte 100 Lt arba jums būtų išduoti pinigai, bet 100 Lt dar nebūtų išskaičiuoti iš jūsų sąskaitos. Tai situacijos, kai viena iš užduočių įvykdoma, o kita – ne. Tikrovėje suma visų pirma nuskaičiuojama iš jūsų sąskaitos ir operacijai nepavykus bei jums negavus pinigų DBVS atkurtų operaciją, atpažindama, kad jūs dar negavote pinigų ir *grąžintų* 100 Lt į jūsų sąskaitą.

DBVS privalo turėti duomenų prieigos valdymo galimybę, kad būtų galima apsaugoti juos nuo piktavalių naudotojų. Tam tikriems vartotojams turi būti leidžiama keisti duomenis, tuo tarpu kiti vartotojai galės tik juos peržiūrėti. Tam skirtas atskirų vartotojų sąskaitų valdymas (vartotojo vardai ir slaptažodžiai) ir teisės (galimybė skaityti arba rašyti konkrečią duomenų bazėje esančią informaciją).

Įsivaizduokite slaptų duomenų rinkinį, pavyzdžiui, žvejų žūklės vietas ir kiekvieną dieną konkrečioje vietovėje sugaunamo laimikio kiekio informaciją. Galime sukurti tris prieigos lygius ir priskirdami naudotojo vardą ir slaptažodį, kad galėtume patikrinti naudotoją, kontroliuoti naudotojo lygį. Šie trys lygiai gali būti:

> Ribota prieiga: šis prieigos lygis tikriausiai bus suteiktas visiems organizacijos nariams, apdorojantiems šiuos duomenis. Šiam lygiui priklausantys asmenys galės peržiūrėti žūklės vietas, bet nematys laimikio kiekio ar konkretaus žvejybos laivo žūklės informacijos.
>
> Teisė tik skaityti: šiam lygiui priklausantys asmenys galės matyti visus duomenis, įskaitant laivų bei laimikio informaciją, bet negalės jų keisti. Šiam lygiui priklausantys asmenys gali būti valdytojai arba mokslininkai, kuriems reikia priimti sprendimus atsižvelgiant į šiuos duomenis (pvz., kvotų paskyras), bet kurie nevaldo šių duomenų.
>
> Visos teisės: šiam prieigos lygiui priklausantys asmenys galės redaguoti informaciją ir pildyti duomenų rinkinį. Dažniausiai tokių asmenų būna labai nedaug ir jie turi konkretų techninį išsilavinimą.

Tokia duomenų struktūra užtikrina, kad slapta informacija pasiekiama tik tiems asmenims, kuriems jos reikia ir tik nedaugelis gali keisti duomenis. Bendra informacija, kurioje nėra jokių slaptų duomenų, būtų pasiekiama didelei naudotojų bendruomenei. Naudotojai, negalintys įrodyti savo tapatybės naudodami naudotojo vardą ir slaptažodį, visiškai neturi prieigos prie duomenų.

Jei naudotojams sudėtinga pasiekti arba keisti duomenis, sistema tampa neefektyvi. DBVS sąsaja turi būti aiški, o duomenys valdomi naudojant paprastas komandas. Tada duomenys gali būti *patogiai* pasiekiami. Daugelis duomenų bazių valdymo programų turi grafines naudotojo sąsajas (GUI), kokias esate įpratę matyti Windows programose, kuriose duomenys valdomi naudojant pelę ir klaviatūrą. Jose naudojamas įprastas duomenų bazės valdymo įvedant tam tikras komandas būdas. Tai vadinama **struktūrizuota užklausų kalba** arba SQL. Daugiau apie ją sužinosite vėliau šio kurso metu.

Turi būti teikiama *efektyvi* prieiga prie DBVS duomenų. DBVS turi veikti efektyviai, o tai reiškia, kad turi būti užtikrinama efektyvi prieiga prie visų veiksmų: nuo duomenų lentelių sisteminimo būdo iki fizinio failų įrašymo į diską. DBVS dar naudojami efektyvūs duomenų ieškos būdai. Pavyzdžiui, kad galėtų pateikti vienos banko sąskaitos balansą, sistemos duomenų neieško tikrindamos visus failus iš eilės.

1.1.2 Duomenų bazių modeliai

Pirmosios kompiuterių programos saugojo visus duomenis failuose ir failų aplankuose, taip, kaip dabar tvarkote teksto apdorojimo failus. Greitai paaiškėjo tokio tvarkymo trūkumai: norint gauti informaciją iš kelių failų, gali prireikti žinoti, kur ir kaip saugoma tokia informacija. Be to, išgavimas labai ilgai užtruktų. Norint sėkmingai saugoti failus, reikia daug programuoti, be to, sunku valdyti saugos funkcijas. Padidėjo sisteminio informacijos saugojimo ir valdymo būdo poreikis, nes padidėjo duomenų apimtis ir sudėtingumas.

Atsiradus geresnio informacijos valdymo būdo poreikiui, buvo sukurtos kelios teorijos. Kiekviena jų buvo sukurta pagal ankstesnes teorijas ir kiekviena jų buvo siekiama panaikinti ankstesnių teorijų apribojimus. Trumpai apžvelgsime kiekvieną konceptualų modelį ir ištirsime stipriąsias bei silpnąsias jo vietas.

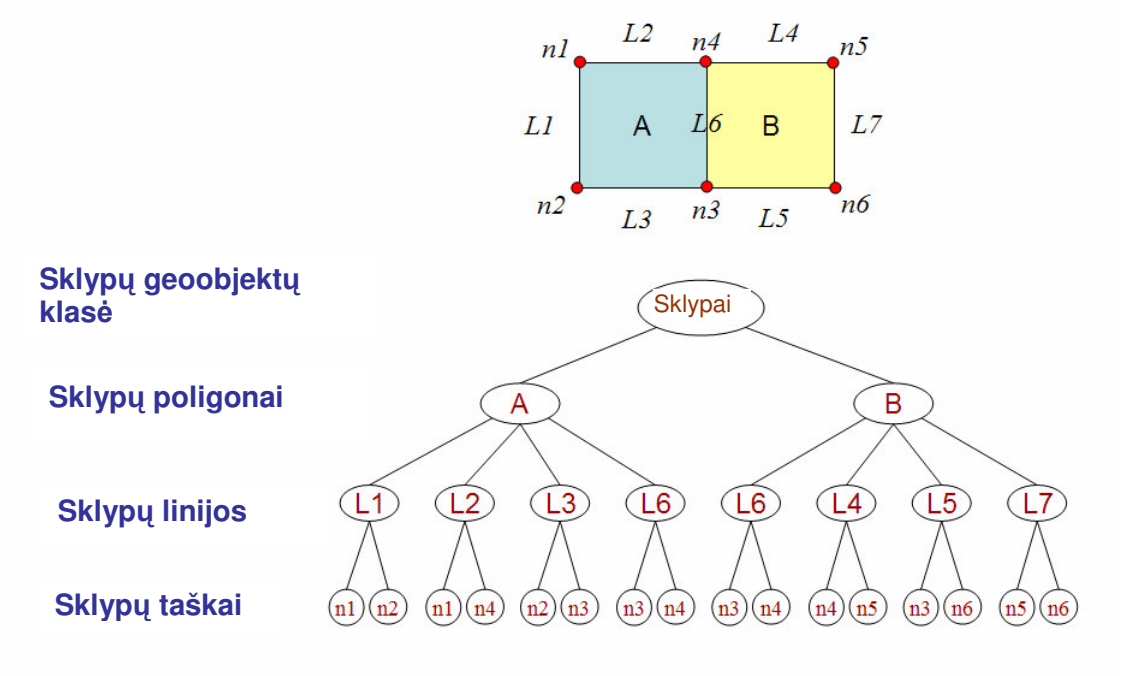
<u>Hierarchinis modelis</u>

Hierarchiniame duomenų bazės modelyje duomenų bazės elementai saugomi kaip medžio struktūros tipas. Terminas *medis* apibūdina hierarchinę diagramą, kuri atrodo kaip apverstas medis. Viršuje yra medžio šaknis, o pačioje – lapai. Apibūdinant hierarchinės struktūros dalis dažnai vartojami terminai *šaknis* ir *lapas*. Šio tipo duomenų bazėje duomenų bazės elementai jungiami ryšių rinkiniu, taip vadinamu pirminių ir antrinių (*parent-child*) įrašų ryšiu. Kiekvienam pirminiam įrašui gali priklausyti vienas ar daugiau antrinių įrašų. Šis ryšių tipas vadinamas vienas su daugeliu (sutrumpintai 1:M arba 1-M). Hierarchinio ryšio pavyzdžiai:

- Biologinis klasifikavimas: pvz., kiekvieną gentį sudaro daug rūšių
- Poligonas – linija – taškas: pvz., kiekvieną liniją sudaro daug taškų

1-2 pav.1–2 pav. Hierarchinės duomenų bazės modelis

pateiktas hierarchinių struktūrų, naudojamų vaizduojant žemės sklypų rinkinį, pavyzdys. Paveiksle viršuje parodyti duomenų bazėje atvaizduoti elementai: dviejų žemės sklypų poligonai (A ir B), septynios linijos (nuo L1 iki L7) ir šeši mazgai arba taškų elementai (nuo n1 iki n6). Poligonai formuojami naudojant uždaras linijas. Kiekviena jų prasideda pradžios tašku ir baigiasi pabaigos tašku. Toliau parodyti jungiamųjų linijų ryšiai naudojant medžio struktūrą. A sklypą sudaro L1, L2, L3 ir L6 linijos. 1 linija jungia n1 ir n2 taškus.



1–2 pav. Hierarchinės duomenų bazės modelis

Didžiausias šio hierarchinio modelio pranašumas buvo tas, kad tai buvo pirmasis bandymas struktūrizuoti duomenis, o ne tik naudoti failų ir aplankų struktūras tvarkant didelius duomenų kiekius. Rezultate DBVS programa gali valdyti duomenų saugą ir vientisumą kompleksiniu būdu.

Dar vienas hierarchinės struktūros pranašumas yra greita prieiga prie didelių duomenų rinkinių. Pavyzdžiui, norint rasti taškinius elementus, kurie formuoja A sklypo perimetrą pirmiau patektame pavyzdyje, duomenų bazės paieška paprasčiausiai randa šaknies ryšius su A sklypu, jo antriniais įrašais (L1, L2, L3 ir L6) ir jų sujungimo taškus. Mums nereikia peržiūrėti dešinės šios diagramos pusės. Nestruktūrizuotame taškų sąraše tokia paieška galėtų apimti kiekvieno duomenų bazėje esančio taško elemento nuskaitymą. Hierarchinis modelis gali labai sumažinti nuskaitymų, reikiamų ieškant duomenų bazės elemento, skaičių.

Tačiau hierarchinis modelis turi kelis trūkumus. Pagrindinis trūkumas yra tas, kad ryšius galima kurti tik vertikaliai, o ne horizontaliai ar įstrižai. Tai reiškia, kad nėra ryšio tarp to paties lygio elementų, nebent jie priklauso tam pačiam pirminiam įrašui. Todėl daugelį duomenų bazės įrašų reikia pakartoti.

Pirmiau pateiktame pavyzdyje, kuriame poligonai priklauso tai pačiai linijai (pvz., L6), reikia pakartoti duomenų modelio elementą, antrinis įrašas negali būti siejamas su daugiau negu

vienu pirminiu įrašu (pvz., L6 negali būti siejama su A ir B poligonais). Taškų elementą reikia pakartoti kiekvieną kartą kai jis panaudojamas kitos linijos elemente. Nors diagramoje yra tik šeši taškai, hierarchiniame modelyje turime išsaugoti 16 taškų elementų.

Reikia atkreipti dėmesį, kad hierarchiniame modelyje trūksta **struktūrinės nepriklausomybės**. Struktūrinė nepriklausomybė atsiranda tada, kai duomenų saugojimo būdai neįtakoja to, kaip DBVS programinės įranga valdo duomenis. Naudojant hierarchinį modelį ir pakeitus duomenų saugojimo būdą reikia pakeisti visas taikomąsias programas, kurios turi prieigą prie duomenų bazės. Todėl duomenų bazės ir programų valdymas gali būti labai sudėtingas ir pareikalauti daug laiko.

Be to, egzistuoja daug realių ryšių, kurių negalima modeliuoti naudojant griežtą „vienas su daugeliu“ ryšių hierarchiją. Įsivaizduokite ryšius bibliotekoje, kai yra knygų ir skolininkų rinkiniai. Bet kurią knygą gali pasiskolinti daug skirtingų žmonių, o tam tikras žmogus gali pasiskolinti neribotą skaičių knygų. Ši situacija negali būti apibūdinama pirminio ir antrinio įrašų ryšiu, todėl šis pavyzdys dažnai vadinamas ryšiu „daugelis su daugeliu“ (sutrumpintai M:M arba M:N).

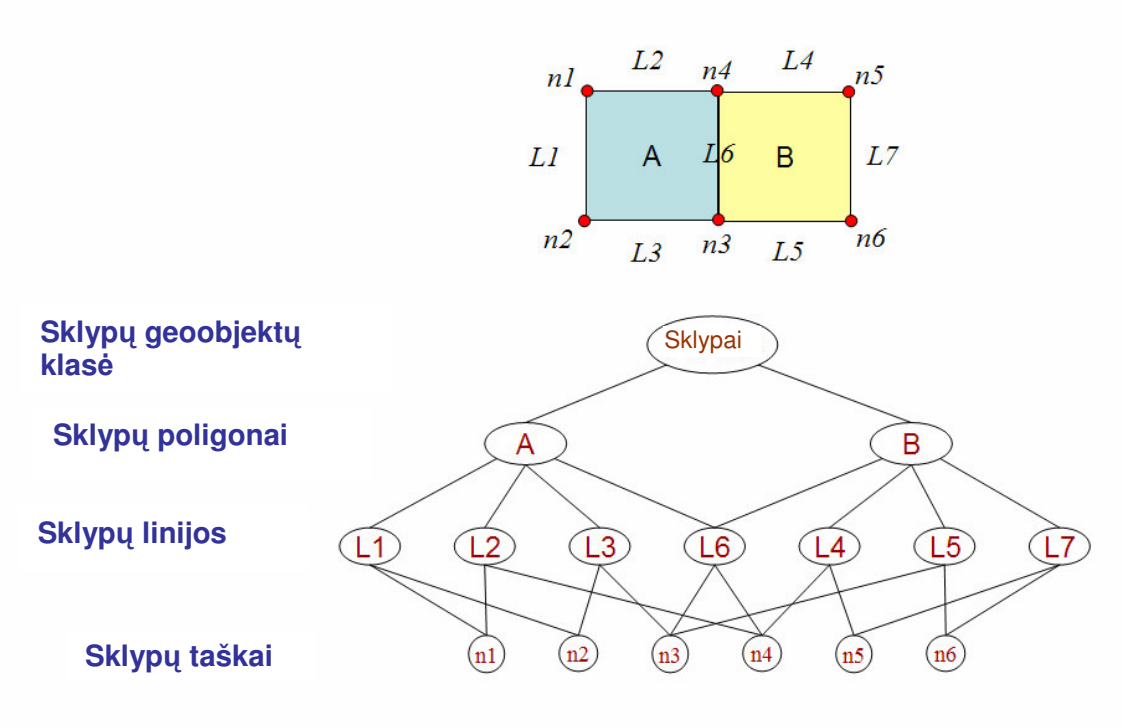
Hierarchinės DBVS buvo populiarios nuo septintojo dešimtmečio pabaigos, kai buvo įdiegta IBM informacijos valdymo sistemų (IMS) DBVS, aštuntame dešimtmetyje ir devintojo dešimtmečio pradžioje.

Tinklo modelis

Tam tikri duomenys modeliuojami natūraliau. Pavyzdžiui, vienas antrinis įrašas priklauso daugiau negu vienam pirminiam įrašui arba ryšiais yra susiję antriniai įrašai, kurie nepriklauso vienam pirminiam įrašui. Šiems ryšiams įtvirtinti buvo sukurtas tinklo modelis. Skirtingai negu hierarchinėse duomenų struktūros, tinklo modeliai nėra apriboti vertikaliais pirminio ir antrinio įrašų ryšiais. Modelio ryšiai gali būti vertikalūs, horizontalūs ir įstriži.

1-3 pav. pavaizduotas to paties sklypų rinkinio, kurį modeliavome naudodami hierarchinę struktūrą, tinklo realizavimo pavyzdys.

Didžiausias skirtumas nuo hierarchinės struktūros yra tas, kad naudojama mažiau duomenų bazės elementų, bet egzistuoja daug daugiau ryšių tarp elementų. Atkreipkite dėmesį, kad linija L6 priklauso dviem pagrindiniams įrašams, A sklypas ir B sklypas turi ryšį su elementu L6. Taškas n3 yra susietas trimis linijomis (L3, L5 ir L6), kurios susikerta šiame taške.



1-3 pav. Tinklo duomenų bazės modelis

Tinklo modelio pranašumas yra jos paprastumas ir daug didesnis pritaikomumas. Naudojant šį modelį galima modeliuoti daugiau realiųjų ryšių negu naudojant griežtą hierarchiją. Be to, šiame modelyje pradedama įgyvendinti atitikimo standartams idėja. Buvo standartizuota visų tinklo modelio realizavimų duomenų bazės sąveikos kalba ir taip palengvintas duomenų bazės administravimas bei mobilumas.

Didžiausias tinklo modelio trūkumas yra sistemos sudėtingumas, nes egzistuoja daug valdomų duomenų bazės elementų ryšių. Norėdami pasiekti elementus duomenų bazės vartotojai turi gerai suprasti duomenų struktūrą, nes prieiga prie elementų naudojama atsižvelgiant į struktūros ryšius. Be to, trūksta struktūrinės nepriklausomybės. Kaip ir hierarchiniame modelyje, pakeitus struktūrą reikia keisti taikomąsias programas.

1971 m. Duomenų sistemų kalbų konferencijoje (CODASYL) buvo oficialiai apibrėžtas tinklo modelis, bet devintame dešimtmetyje jį beveik visur pakeitė reliacinis (sąryšinis) modelis.

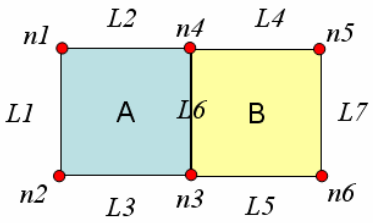
Reliacinis (sąryšinis) modelis

Reliacinio modelio pagrindas yra **santykiai** (lentelė). Jie tvarko ryšius tarp daugelio duomenų bazės lentelių. Šis modelis pagrįstas matematikos sričiai priklausančia santykine algebra. Reliacinį modelį išsamiau aptarsime vėliau. Reliaciniame modelyje informacija vaizduojama kaip lentelių rinkiniai, kuriose eilutės atstoja duomenų elementus. Ryšius tarp lentelių valdo reliacinių duomenų bazių valdymo sistemos (RDBVS) programinė įranga. Šią lentelių struktūrą vartotojai nesunkiai supranta, be to, su ja nesudėtinga dirbti.

1-4 pav. paveikslėlyje parodyta, kaip naudojant reliacinę duomenų bazę galima atvaizduoti žemės sklypus.

Lentelė. Poligonai

Poligona	Linijos
A	L1, L2, L3, L6
B	L6, L4, L7, L5



Lentelė. Linijos

Linija	Taškas	Taškas
L1	n1	n2
L2	n2	n4
L3	n2	n3
L4	n4	n5
L5	n3	n6
L6	n3	n4
L7	n5	n6

Lentelė. Taškai

Taškas	X ašis	Y ašis
n1	x_1	y_1
n2	x_2	y_2
n3	x_3	y_3
n4	x_4	y_4
n5	x_5	y_5
n6	x_6	y_6

1–4 pav. Reliacinis modelis

Pagrindinis reliacinio modelio pranašumas yra struktūrinė nepriklausomybė. Šiame modelyje vidinių duomenų struktūrų pakeitimai nedaro įtakos DBVS duomenų prieigai. Naudojant reliacinį modelį vartotojui nereikia išmanyti fizinės duomenų bazės struktūros. Dar vienas reliacinio modelio pranašumas yra sąlyginai paprasta struktūra (ir sąlyginai paprastas projektavimas bei pritaikymas) ir galimybė duomenų bazėse vykdyti užklausas naudojant standartinę **struktūrizuotą užklausų kalbą** (SQL).

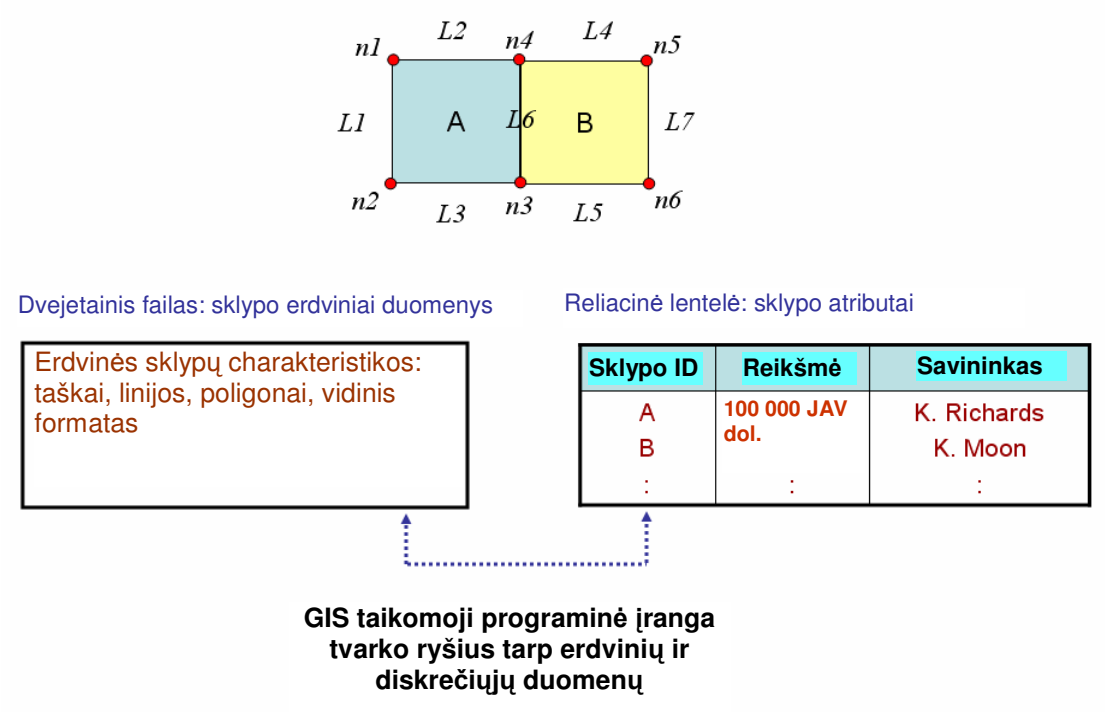
Tačiau sąlyginis reliacinio modelio paprastumas turi savo kainą. Vienas iš pagrindinių reliacinio modelio trūkumų yra didelis techninės ir programinės įrangos resursų naudojimas. Tvarkant ryšius tarp elementų daug sudėtingų ryšių paslepiama nuo vartotojo, todėl norėdama gauti prieigą prie duomenų ir juos tvarkyti reliacinės duomenų bazės valdymo sistema (RDBVS) turi įvykdyti daug užduočių. Tiesą sakant, kai 1970 m. E.F. Codd pasiūlė reliacinį modelį, kompiuteriai dėl galios trūkumo negalėjo apdoroti šio modelio (Rob ir Coronel, 2000). Dabar, padidėjus kompiuterių duomenų apdorojimo spartai, yra daug efektyvių reliacinio modelio pritaikymo būdų, bet reliacinės duomenų bazės vis dar yra lėtesnės negu kitos duomenų bazių sistemos.

Reliacinė struktūra yra populiarus GIS modelis. Pavyzdžiui, toliau pateikti plačiai naudojami reliacinių duomenų bazių taikymai:

- INFO, sistemoje ARC/INFO;
- *MS Access*, skirta *ArcMap*;
- *dBASE III*, skirta PC kompiuteriuose veikiančioms GIS;
- *ORACLE, SQL Server*, įvairioms GIS.

Georeliacinė struktūra

Nors tam tikros GIS taikomosios programos visų duomenų tvarkymui naudoja RDBVS (pvz., ESRI geoduomenų bazės struktūra), anksčiau buvo įprasta erdvinius ir atributinius tam tikros geoobjektų klasės duomenis saugoti naudojant atskiras struktūras. Erdviniai duomenys (taškai, linijos ir poligonai) buvo saugomi tam tikru dvejetainiu formatu, o geoobjektų atributai (pvz., sklypo savininkas ar vamzdžio skersmuo) buvo saugomi standartiniu RDBVS formatu, tokiu kaip dBASE ar INFO. Tada ryšį tarp šių dviejų failų struktūrų tvarkydavo GIS programinės įrangos taikomosios programos, užtikrindamos, kad taikomi reikiami tam tikrų erdvinių elementų atributai. Šis duomenų skirstymo į du formatus metodas vadinamas **Georeliacine struktūra**, arba **Dvejopa architektūra**. 1-5 pav. pavaizduota, kaip gali būti struktūrizuojami mūsų plotų duomenys naudojant georeliacinę struktūrą.



1-5 pav. Georeliacinė struktūra

Ši georeliacinė struktūra nėra toks duomenų modelis, kokį naudojome aptardami hierarchinius, tinklo ar reliacinius modelius. Tai struktūra, kuri pritaiko reliacinį modelį siekdama konkretaus tikslo.

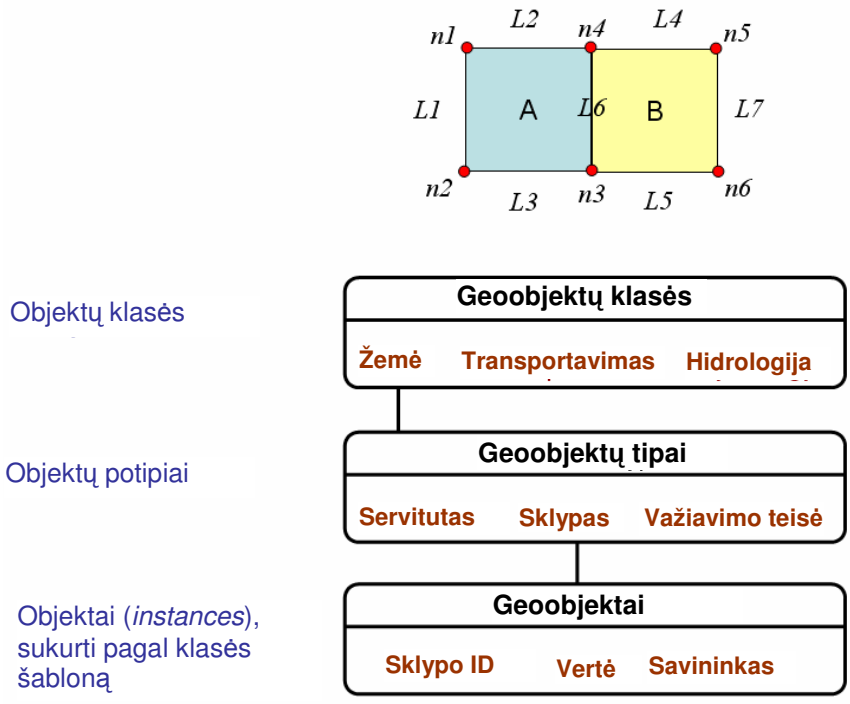
Nors besivystant GIS atsirado integruotų geoduomenų bazių, kuriose yra visi reikiami duomenys, dažnai vis dar naudojami dvejopos architektūros modeliai. GIS, kuriuose naudojama dvejopa architektūra, pavyzdžiai: Arc/INFO, MapInfo, ArcView ir MGE.

Objektinis modelis

Noras efektyviau atvaizduoti realiojo pasaulio reiškinius ir ryšius skatino sukurti naują, lankstesnį ir sudėtingesnį duomenų bazės modelį. 1981 m. buvo sukurtas semantinių duomenų modelis (SDM). Jame pirmą kartą buvo paminėta sąvoka **objektas**. Žodis *semantinis* nurodo reikšmę. Į semantinį modelį, naudojant vieną struktūrą, norėta įtraukti platesnį supratimą apie daiktus, įskaitant jų elgseną ir ryšius.

Esami objektiniai duomenų bazių modeliai (ODBM), kurie buvo sukurti atsižvelgiant į SDM, apima pagrindinius duomenų bazės elementus ir jų atributus (kurie yra ir reliaciniame modelyje). Jos dar apima ryšius tarp atributų *pačiame* objekte ir visas operacijas, kurios gali būti atliktos naudojant kiekvieną objektą arba būti atliktos kiekvieno objekto, pvz., objekto vertės pakeitimas, konkrečios vertės radimas ar vertės parodymas. Sutraukiant ryšius, duomenis ir operacijas į vieną objektą sukuriamas savarankiškas objektas. Objektas tampa kūrimo bloku, kurį galima bendrai naudoti, perkurti ar panaudoti kurti sudėtingesnius objektus.

1-6 pav. parodyta, kaip gali atrodyti objektiniame modelyje saugomi sklypų duomenys.



1-6 pav. Objektinis modelis

Pagrindinė mintis yra ta, kad vartotojams nereikia nagrinėti eilučių ir stulpelių, o kreipti dėmesį į objektus ir tų objektų operacijas, kurios labiau primena realiame pasaulyje egzistuojančius dalykus. Pavyzdžiui, vietoje to, kad įterptumėte darbuotojo įrašus darbuotojų lentelėje, sukurtumėte ryšius su skyriaus lentele ir t. t., jūs naudojate operaciją *samdyti*. Naudodami objektiškai orientuotas (OO) struktūras, sudėtiniuose objektuose įtvirtiname „sumanumą“. Labai panašiai kaip ir geografiniuose objektuose ir ryšiuose, kurie gali būti pakankamai sudėtingi.

Kaip ir reliacinis modelis, objektinis modelis išsaugo struktūrinį vientisumą, todėl galima keisti duomenų struktūrą nekeičiant programinės įrangos ar nuskaitymo procedūrų. Didžiausias modelio pranašumas yra galimybė pridėti semantinį turinį ir suteikti duomenims daugiau prasmės.

Didžiausias objektinių modelių trūkumas yra standartų stoka, pvz., reliacinių modelių pagrindinis duomenų prieigos metodų standartas yra SQL. Objektinių struktūrų duomenų prieigos metodai gali būti sudėtingi. Juos galima palyginti su hierarchinių arba tinklo modelių duomenų prieigos metodais. Objektinių modelių sudėtingumas reikalauja apdoroti daug duomenų bazės sąveikų, todėl tam tikrų pritaikymų operacijos vykdomos daug lėčiau negu reliacinėse struktūrose.

Tinklo ir hierarchiniai modeliai vis dar naudojami, bet jau nebėra vyraujantys. Galima sakyti, kad šie modeliai yra techniškai pasenę (2000 m.). Didžioji dauguma dabar naudojamų sistemų yra reliacinės. Didelės investicijos į reliacines technologijas apsunkino perėjimą prie objektinių sistemų. Daugelyje dabar naudojamų reliacinių panaudojimų buvo pritaikyti objektiniai principai. 1999 m. nemažai šių principų buvo įtraukta į standartinę SQL.

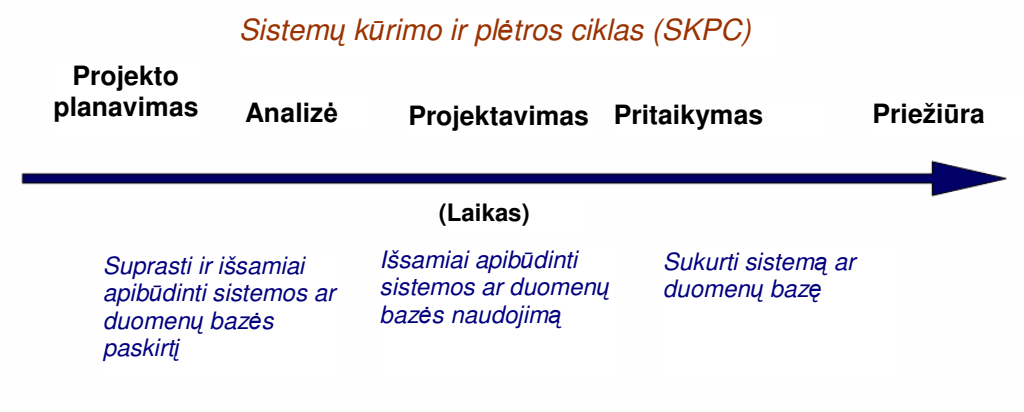
Šie reliaciniai panaudojimai su pridėtomis objektinėmis galimybėmis (pvz., pasirinktinių objektų tipų nustatymas) kartais vadinami **objektiniu/reliaciniu modeliu**. Tačiau duomenų lygiu tai dažniausiai yra reliacinė struktūra, turinti objektinių funkcijų, kurios pridedamos programinės įrangos RDBVS lygiu. Tvirtinama, kad daugelis didžiųjų RDBVS programų (pvz., Oracle, DB2, Informix, SQL Server) palaiko objektines technologijas įvairiu lygiu. Tikimasi, kad reliacinės ir objektinės technologijos bus suliejamos ir ateityje (Rob ir Coronel, 2000 m.).

1.2 Duomenų modeliavimas

Šioje temoje išsiaiškinsime duomenų bazių projektavimo aspektus. Projektavimas yra realaus pasaulio modeliavimas arba referavimas į konkrečią, geriausiai duomenų bazei tinkančią, struktūrą. Sudarant duomenų bazę yra svarbi duomenų struktūra ir aiškus supratimas, kaip duomenų bazė bus naudojama. Tik tada galima sukurti naudingą ir sėkmingą duomenų bazės modelį.

Kurdami taikomąsias programinės įrangos ar duomenų bazės programas vadovaujamės sistemų kūrimo ir plėtros ciklu **(SKPC). Pirmiausia reikia nustatyti sistemos ar duomenų bazės paskirtį. Projektavimo fazė yra tokia fazė, kai dokumentais pagrindžiame, kaip sistemoje bus įgyvendinami šie reikalavimai. Šioje fazėje reikia pagrįsti techninės ir programinės įrangos reikalavimus, programinės įrangos ar duomenų bazės specifikacijas, mokymų reikalavimus ir duomenų perkėlimo planus. Pritaikymas yra duomenų bazės sukūrimas, techninės ir programinės įrangos diegimas, darbuotojų apmokymas ir esamų duomenų perkėlimas į naujas struktūras. Priežiūra apima sistemos tobulinimus, klaidų taisymus ir pagalbą vartotojams ar jų mokymus. 1-7 pav.1-7 pav. Sistemų kūrimo ir plėtros ciklas**

šis ciklas pavaizduotas grafiškai.

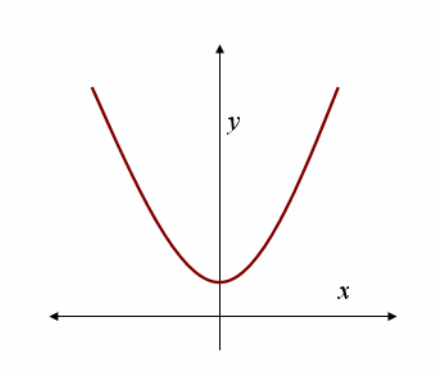


1-7 pav. Sistemų kūrimo ir plėtros ciklas

Diegdami projektą su konkrečiais GIS reikalavimais projektavimo etapu turime atsižvelgti į dar vieną dalyką: kaip duomenų bazės elementai bus atvaizduojami erdviškai. Reikia nuspręsti, kas mūsų duomenų bazėje bus vaizduojama kaip taško, linijos ar poligono elementų klasės, kokiu masteliu bus fiksuojami duomenys ir kokių rūšių ryšiai galės būti naudojami.

1.2.1 Duomenų modeliai

Modelis yra sistemą ar sistemos pogrupį apibūdinanti abstrakcija. Pavyzdžiui, 1-8 pav. diagrama gali vaizduoti sviedinio trajektorija arba išreikšti ekonominę veiklą. Akivaizdu, kad tikrovėje bus subtilybių ir netikslumų, kurių nebus galima apibūdinti modelyje, bet galime žiūrėti į tai kaip į paprastą fizikos ar ekonomikos sričių veiksnių atvaizdavimą.



1-8 pav. Modelio pavyzdys

Žymėjimas yra grafinių ar tekstinių abstrakcijos vaizdavimo taisyklių rinkinys. Pavyzdžiui, pirmiau pavaizduotą diagramą galima atvaizduoti matematine lygtimi:

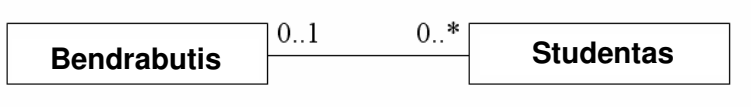
$$y = x^2 + 2$$

Matematikams žymėjimo taisyklių rinkinys nurodo, kad x^2 reiškia x vertės pakėlimą antruoju laipsniu. Be to, + simbolis nurodo, kad prie x^2 vertės reikia pridėti 2. Žymėjimo taisyklės leidžia interpretuoti modelyje naudojamus matematinius simbolius.

Sistemos yra sudėtingos ir jas sunku suprasti. Tam tikri aspektai modeliuose matomi aiškiau negu realioje sistemoje. Pavyzdžiui, gali būti sunku suprasti viską, kas vyksta labai sudėtingoje duomenų bazės sistemoje. Tačiau duomenų struktūros dokumentavimas naudojant diagramas suteikia skaitytojui galimybę išskirti tam tikrus aspektus ir išsamiai juos ištirti. Duomenų bazės modelis suteikia projektuotojui galimybę:

- Išreikšti savo mintis ir bendrauti su kitais žmonėmis
- Pagrįsti sistemą:
  - Aptikti klaidas
  - Numatyti kokybę
- Naudojant kompiuterio padedamos programų inžinerijos (CASE) programinę įrangą galima generuoti realios sistemos dalis:
  - Programavimo kodą, kad būtų galima valdyti funkcionavimą
  - Duomenų struktūras (lenteles ir laukus), kurias naudojant bus formuojama mūsų galutinė duomenų bazė

Loginis modelis yra atliekamų veiksmų aukšto lygio abstrakcija. Atsižvelgiant į duomenų bazę, tai yra apibendrintas duomenų bazėje saugomų duomenų aprašymas ir saugojamų duomenų siejimo su kitais duomenimis būdas ir jų funkcionavimo baigtoje sistemoje ar duomenų bazėje, apibūdinimas. Pavyzdžiui, 1-9 pav. diagramoje labai bendrai pavaizduoti duomenų bazėje esančių duomenų ryšiai. Šioje duomenų bazėje gali būti saugoma informacija apie studentus ir jų bendrabučius. Vėliau šioje temoje išsamiau aptarsime šioje diagramoje naudojamą žymėjimą.



1-9 pav. Loginio modelio pavyzdys

Fizinis modelis yra praktinis ir išsamus atvaizdavimas. Fizinis modelis nurodo, kaip duomenys bus fiziškai saugomi laikmenose (failų tipai, duomenų saugojimo diskai) ir duomenų struktūrą atsižvelgiant į lentelių eilutes ir stulpelius. Pavyzdžiui, naudojant duomenų apibūdinimo kalbą (*data definition language - DDL*) galima labai konkrečiai apibūdinti, kaip bus kuriama nauja DBVS lentelė. Vėlesnių paskaitų metu aptarsime šios kalbos elementus, pvz., duomenų tipus ir pirminius raktus.

```
CREATE TABLE EMPLOYEE (
        EMP_ID NUMBER(8),
        LNAME VARCHAR2(30),
        FNAME VARCHAR2(15),
        HIRE_DT DATE,
        SALARY NUMBER(8,2)
);

ALTER TABLE EMPLOYEE (
        CONSTRAINT EMP_PK
        PRIMARY KEY (EMP_ID)
);
```

Šios temos paskaitų metu pagrindinis dėmesys bus skiriamas aukštesnio lygio loginiam modeliavimui. Perėjimo nuo loginio modelio prie fizinio modelio procesą aptarsime vėlesnėse šio kurso dalyse.

1.2.2 Klasių diagramos

Vieninga modeliavimo kalba (UML) yra žymėjimas, naudojamas apibūdinant programinę įrangą ir duomenų bazes. Jis apima diagramas, kuriose apibūdinama įvairi informacija, pvz., kaip vartotojas bendrauja su programine įranga. Daugelis šių dalykų nebūtinai įtakoja duomenų struktūrą.

Iki 1997 m. buvo naudojamas bendras žymėjimas, vadinamas elementų ryšių diagrama (ERD). Vis dar galite dažnai susidurti su šio žymėjimo terminija, todėl kur galima įtrauksime UML ir ERD terminus. Aptardami šią medžiagą užduotyse ar egzaminų metu galėsite vartoti bet kurį iš šių dviejų terminų rinkinių.

UML sudaro 12 diagramų tipų, suskirstytų į 3 kategorijų sritis:

Struktūrinės	Elgesio	Modelių valdymo
1. **Klasių diagrama** 2. Objektų diagrama 3. Komponentų diagrama 4. Diegimo diagrama	5. Naudojimo diagrama 6. Sekos diagrama 7. Veikimo diagrama 8. Kolaboravimo diagrama 9. Būsenos diagrama	10. Paketai 11. Posistemės 12. Modeliai

Pagrindinį dėmesį skirsime **klasių diagramoms** ir išmoksime apibūdinti duomenų struktūrą. Klasių diagramos apibrėžia duomenų klases ir ryšius, kurie galios baigtoje sistemoje. Analizės fazė dažniausiai pradedama sukuriant apibendrintą klasių diagramą.

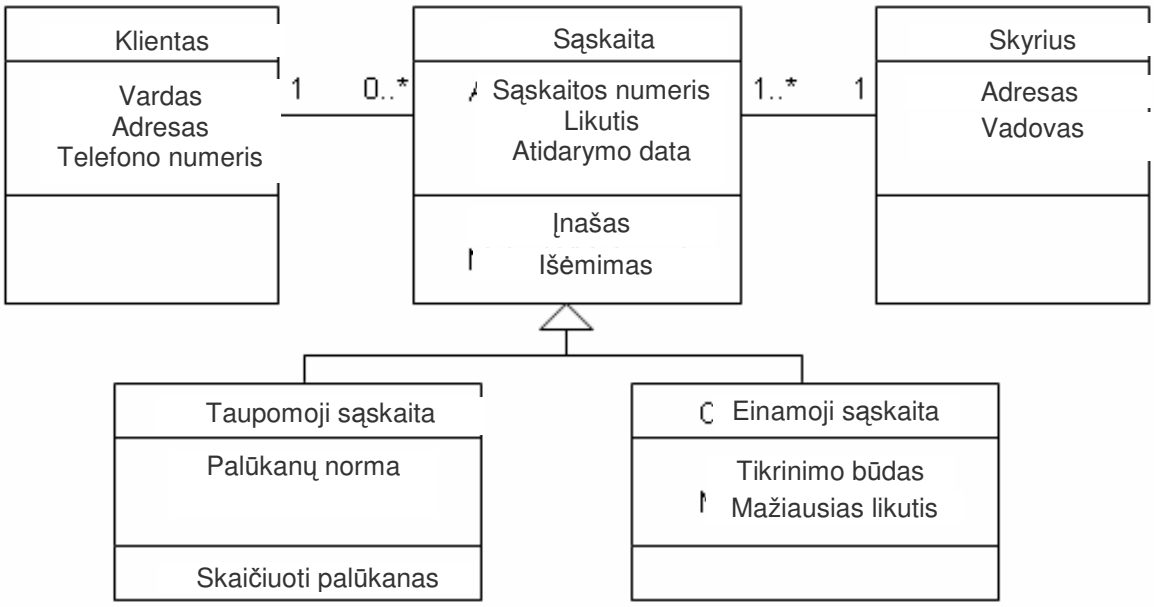
Objektas yra taikymui reikšminga sąvoka, abstrakcija arba daiktas. Pavyzdžiui:

- John Smith, darbuotojas
- Medis, kurį reikia pasodinti
- Žemės sklypas, kuriame reikia atsodinti mišką

Klasės yra objektų, turinčių tokias pačias savybes ar atributus, grupės. Objektas yra objektų klasės vienetas (*instance*). Pavyzdžiui:

- Darbuotojas John Smith yra objektas sukurtas pagal objektų klasės apibrėžimą, visi darbuotojai sudaro objektų *klasę*

Klasių diagramose naudodami žymėjimą įvardijame objektus, jų atributus ir objektų klasių ryšius. 1-10 pav. parodytas UML klasių diagramos pavyzdys. Išsamiai panagrinėsime šią diagramą, aptarsime skirtingų žymėjimo dalių reikšmes ir diagramos teikiamą informaciją.

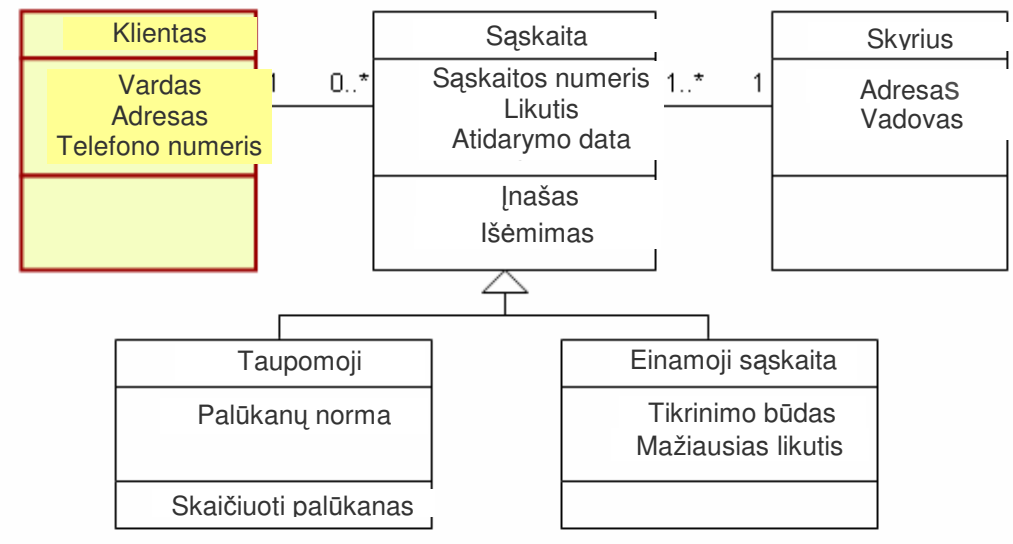


1-10 pav. Klasių diagramos pavyzdys

Čia pavaizduota paprasta bankininkystės duomenų bazės klasių diagrama. Joje paminėti *klientai*, *banko sąskaitos* ir *banko skyriai*. Peržiūrėję diagramą galime suprasti, kad yra dviejų tipų sąskaitos: einamoji ir taupomoji. Diagramoje nurodyta, kad yra ryšys tarp klientų ir jų bankų sąskaitų bei ryšys tarp bankų skyrių ir juose laikomų sąskaitų.

Klasės

Pagrindinės žymėjimas klasių diagramose yra objektų klasės. Jos diagramoje yra vaizduojamos langeliais. Klientų klasė paryškinta 1-11 pav.



1-11 pav. Klasių diagramos klasės

Klasių langeliai suskirstyti į tris skyrius:

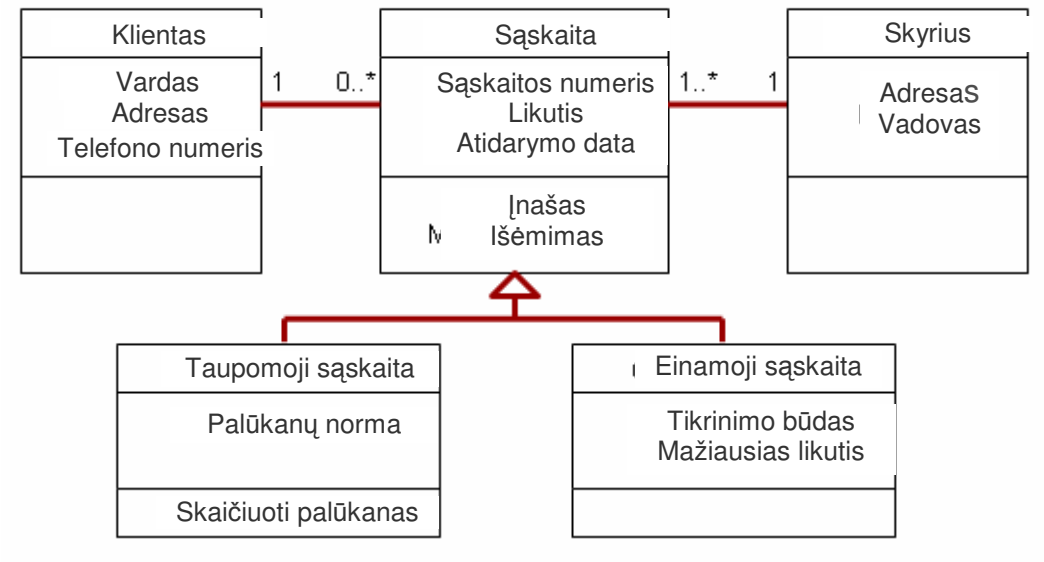
- viršutiniame skyriuje nurodytas klasės pavadinimas;
- viduriniame skyriuje yra objekto **ypatybės** arba atributai;
- apatiniame skyriuje yra **metodai** arba veiksmai, kuriuos atlieka klasė arba kurie yra atliekami su ja.

Jeigu esate susidūrę su objektiniu programavimu, terminai objektas, ypatybė ir metodas jums jau žinomi. ERD žymėjime objektų klasės sąvoką atitinka **būtis** (entity). Klasių diagramos pagrindiniai dalykai yra objektų klasės, o būsenų-ryšių diagramoje – vienetai.

Tikriausiai jau girdėjote GIS terminą **atributas**. Fizinėje duomenų bazėje jie atitinka geoobjektų atributų lentelės stulpelius. Mes vengsime nagrinėti metodus. Apskritai, metodai apibūdina duomenų bazių savybes, kurios verčiamos į programuotojų numatytus duomenų bazės veiksmus. Pradėję naudoti ESRI geoduomenų bazę pamatysime, kad galima naudoti savybes – metodus – kuriuos programuotojai jau įtraukė į duomenų bazę.

Asociacijos

Asociacijos yra ryšiai tarp modelio objektų klasių. Bankininkystės pavyzdyje parodyta asociacija tarp banko sąskaitos ir skyriaus, bei asociacija, tarp banko sąskaitos ir kliento, kuriam priklauso ta sąskaita. Klasių diagramoje asociacijos nurodomos linija tarp klasių. 1-12 pav. parodyta bankininkystės diagrama su paryškintomis asociacijomis.



1-12 pav. Klasių diagramos asociacijos

Diagramoje parodyta klientų ir jų banko sąskaitų, bei sąskaitos ir banko skyrių, kuriuose atidarytos sąskaitos, asociacijos. Dar yra ypatingas ryšys tarp sąskaitos ir dviejų bankų sąskaitų tipų (taupomosios ir einamosios). Tai paveldimumo asociacija, ją aptarsime šiame skyriuje.

Asociacijos dažnai pavadinamos. Pavyzdžiui, šalia klientų ir sąskaitų asociacijos galima įterpti žodį „valdo“ ir parodyti, kad klientas yra sąskaitos savininkas. Šios pavadintos asociacijos padeda skaityti ir suprasti diagramą.

ERD žymėjime ryšys tarp elementų vadinamas **ryšiu**, BML žymėjime, šis ryšys vadinamas **asociacija**.

Asociacijos gali būti kelių tipų. Paprasčiausios ir dažniausiai naudojamos žymimos nubrėžiant liniją tarp dviejų elementų, kaip parodyta kliento ir jo sąskaitos asociacijoje. Trys sudėtingesnių asociacijų tipai yra:

- Agregavimas
- Komponavimas
- Paveldimumas

Agregavimą **galima apibūdinti kaip specialią asociaciją, kuri simbolizuoja dalies su visuma ryšį. Visas objektas sudarytas iš dalių rinkinių, dalys gali būti prijungiamos prie visumos ir iš jos pašalinamos, dalys gali būti daugiau negu vienos visumos dalimis.T.y. viena klasė gali būti sudaryta iš kelių kitai klasei priklausančių dalių. Pavyzdžiui, skyriaus sąvoka gali apimti daugiau negu jame dirbančius darbuotojus. Agregavimo asociacija parodyta klasių diagramoje naudojant asociacijos linijoje įterptą rombą. Rombas įterptas arčiau „visumos“ klasės. 1-13 pav.1-13 pav. Agregavimo asociacijos pavyzdys**

parodytas tokios asociacijos pavyzdys. Skyrius yra „visuma“, o darbuotojas – „dalis“.

1-13 pav. Agregavimo asociacijos pavyzdys

Tai neapriboti ryšiai todėl, kad skyriui esant visuma, o darbuotojams dalimis, darbuotojas gali pereiti iš vieno skyriaus į kitą, gali nepriklausyti jokiam skyriui arba gali būti susietas su daugiau negu vienu skyriumi.

Automobilis-ratas yra aiškus agregavimo ryšio pavyzdys. Automobiliai yra ne kas kita kaip visų dalių suma. Ratas yra viso objekto (automobilio) dalis, bet galima nuimti ratus nuo vieno automobilio ir sumontuoti ant kito. Nesumontuotas automobilio ratas nėra labai retas reiškinys (pvz., žieminės padangos jūsų garaže).

Komponavimas yra stipresnis ryšys už agregavimą, bet jo esmė labai panaši. Skirtumas yra toks, kad kalbame apie dalies/visumos asociaciją, bet dalys glaudžiau susaistytos su visuma. Kartais jis vadinamas priklausomybės ryšiu. Iš kelių komponavimo ryšio požymių matyti, kad tai yra tikslesnis ryšys negu agregavimo asociacija.

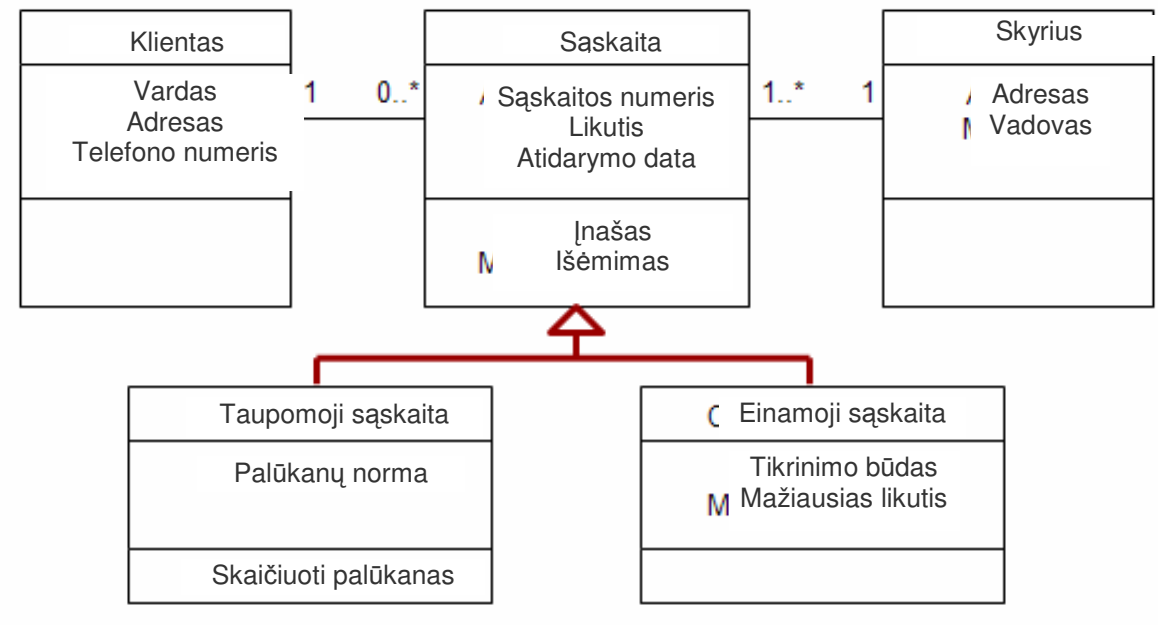
Dalies/visumos komponavime „dalis“ negali egzistuoti atskirai. Galime įsivaizduoti, kad „dalies“ objektai, sukurti „visumos“ objekte niekaip negali būti perkelti į kitą „visumos“ objektą ir panaikinti iš to „visumos“ objekto. Kol kas atmesdami smegenų persodinimo sąvoką galime galvoti apie smegenis kaip apie kūno dalį, kuri buvo sukurta kartu su kūnu ir lieka jame iki mirties. Kūnas yra ne kas kita kaip dalių suma, bet dalys yra labai tiesiogiai susijusios su konkrečiu kūnu.

Komponavimo asociacija parodyta klasės diagramoje naudojant juodą rombą, įterpta asociacijos linijoje. Kaip ir agregavimo asociacijoje, rombas įterptas arčiau „visumos“ objekto. 1-14 pav. parodytas komponavimo asociacijos pavyzdys.

1-14 pav. Komponavimo asociacijos pavyzdys

Šis ryšys sieja kambarius ir bendrabučio pastatus, kuriuose yra kambariai. Pagrindinė mintis yra ta, kad sąvoka „kambarys“ negali egzistuoti jeigu nėra pastato, kuriam fiziškai priklauso tas kambarys. Be to, negalima išimti kambario iš vieno pastato ir įdėti į kitą. Bent jau praktiškai.

Paveldimumo asociacija yra daiktų klasifikavimo priemonė. Gali būti keli *poklasiai*, kurie sudaro *superklasės* apibrėžimą. Poklasio objektai *paveldi* visas superklasės ypatybes ir metodus, be to, jie gali turėti papildomų ypatybių. Klasių diagramoje jie simbolizuojami neužpildytu trikampiu, esančiu šalia asociacijos linijos. Jis yra arčiausiai superklasės ir į ją nukreiptas. Pavyzdžiui, panagrinėkime bankininkystės diagramą. 1-15 pav. parodyta bankininkystės diagrama su paryškinta paveldimumo asociacija.



1-15 pav. Klasių diagramos paveldimumas

Diagramoje parodyta, kad sąskaitos gali būti dviejų tipų: taupomosios ir einamosios. Šiuo atveju superklasė yra sąskaita, poklasiai – taupomosios ir einamosios sąskaitos. Poklasiai, pvz., einamoji sąskaita, bendrai naudoja visus „sąskaitos“ superklasės požymius. Tai reiškia, kad jos turi sąskaitos numerį, likutį ir atidarymo datą. Be to, jos gali turėti kitų požymių, kurie priklauso tiki einamosioms sąskaitoms, pvz., mažiausią mėnesio likutį.

Paukštis-gulbė yra kitas paveldimumo ryšio pavyzdys. Biologinio klasifikavimo schema (rūšis, tipas ir t. t.) parodo, kaip kiekvienas lygis paveldi aukščiau esančio lygio požymius, bet gali turėti tam tikrų naujų požymių. Šiame pavyzdyje gulbė turi visiems paukščiams būdingus požymius (sparnus, snapą ir t. t.), bet papildomi jos požymiai yra dydis ir plėvėtos kojos.

Poklasio–superklasės ar paveldimumo ryšį galite įsivaizduoti kaip „yra“ ryšį. Einamoji sąskaita „yra“ banko sąskaita.

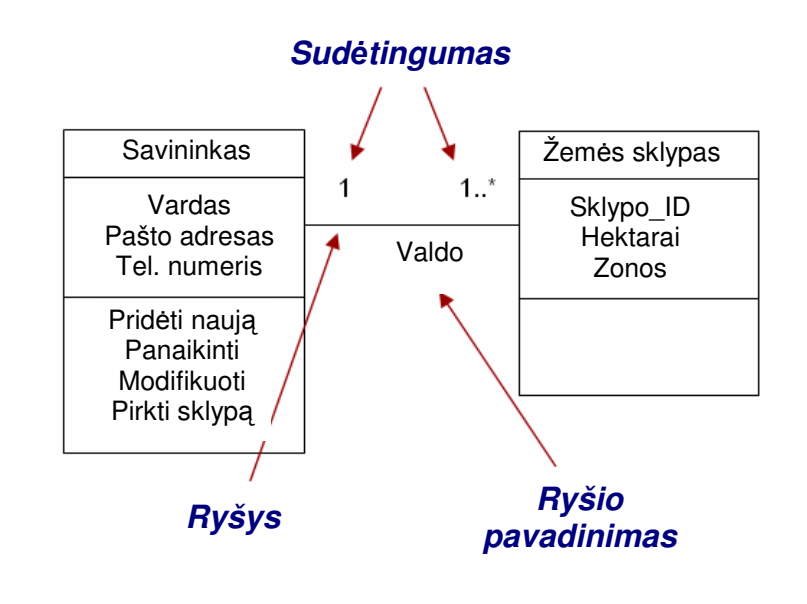
Jei norite klasifikuoti smulkiau, galite naudoti paveldimumo ryšį ir du ar daugiau poklasių. Pavyzdžiui, GIS sistemoje gali būti vandentiekio vamzdžių superklasė, kurią sudaro trys poklasiai: betoniniai vamzdžiai, plieniniai vamzdžiai ir keramikiniai vamzdžiai. Kiekvienas poklasis gali turėti savo atributų.

Terminai „yra“, „dalis“ ir „turi“ gali padėti nustatyti taikomą ryšio tipą.

- Agregavimas – „dalis“
- Paveldimumas – „yra“
- Komponavimas – „turi“

Sudėtingumas

Vaizduodami klasių asociaciją turime nurodyti, kiek vienos klasės objektų yra susiję su kitos klasės objektu. Tą darome nurodydami didžiausią ir mažiausią skaičių. 1-16 pav. parodyta žemės savininkų ir jiems priklausančių žemės sklypų asociacija.



1-16 pav. Klasių diagramos ryšių simbolika

Diagramoje *savininkų* ir *žemės sklypų* ryšys aiškiai pažymėtas naudojant šias dvi klases jungiančią liniją. Reikšmingas pavadinimas „valdo“ nusako mums ryšio pobūdį. Suprantame, kad linija nusako faktą, jog *žemės sklypą* valdo *savininkas*.

Asociacijų sudėtingumo žymėjimai gana painūs. Toliau esančioje lentelėje (1-3 lentelė) nurodytos galimos asociacijų sudėtingumo žymėjimo reikšmės. Asociacijos sudėtingumas turi būti parodytas visose asociacijose, išskyrus paveldimumo asociacijas (atkreipkite dėmesį, kad 1-15 pav. nėra paveldimumo asociacijos sudėtingumo).

1-3 lentelė. Sąsajų sudėtingumo žymėjimas klasių diagramose

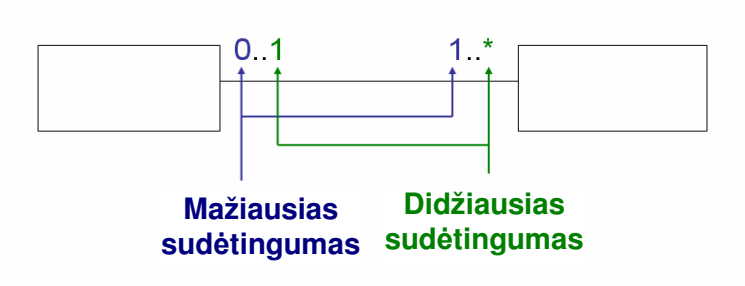
Žymėjimas	Alternatyvi trumpa forma	Aprašas
1..1	1	Tik vienas
0..1		Nulis arba vienas
0..*	*	Nuo nulio iki bet kurio teigiamo sveikojo skaičiaus; nulis arba daugiau
1..*		Nuo vieno iki bet kurio teigiamo sveikojo skaičiaus; vienas arba daugiau
n..m		Bet kuris teigiamas sveikasis skaičius nuo n iki m; daugiau negu vienas

Asociacijos sudėtingumas suprantamas dviem kryptimis, todėl savininko ir žemės sklypo asociacijos linijoje yra du žymėjimai 1-16 pav. Asociacijos kairėje yra žymėjimas „1“, o dešinėje – „1..*“. Jei skaitysime šį ryšį iš kairės į dešinę, turėsime atsižvelgti į dešinėje esantį sudėtingumą („paskirties“ sudėtingumą). Žymėjimą 1..* verčiant į lietuvių kalbą, asociacija informuoja, kad „*Savininkai valdo vieną ar daugiau žemės sklypų* ”. Tai nurodo du dalykus: savininkai turi valdyti bent vieną žemės sklypą ir savininkams gali priklausyti vienas arba daugiau žemės sklypų. Pirmasis apribojimas naudojamas todėl, kad duomenų bazėje nėra prasmės saugoti savininko, jei jam niekas nepriklauso. Antrasis apribojimas taikomas kelių žemės sklypų savininkas. Skaitydami iš dešinės į kairę, atsižvelgsime į kairėje pusėje esantį žymėjimą („1“), kuris reiškia: „*žemės sklypai priklauso vienam ir tik vienam savininkui*“. Atkreipkite dėmesį, kad sutrumpintas žymėjimas „1“ yra toks pats kaip „1..1“. Tai irgi nurodo du dalykus: Žemės sklypas *privalo* turėti savininką ir žemės sklypas negali priklausyti daugiau negu vienam asmeniui.

Asociacijų sudėtingumas nustatomas įvertinant organizacijos darbo arba verslo taisykles, duomenų apdorojimo būdus. Jeigu organizacija nori saugoti savininkus net jiems pardavus paskutinį žemės sklypą (t. y. savininkams žemė nebepriklauso), tada sklypo (dešinėje) sudėtingumas negali būti 1..*, nes 1..* nurodo, kad savininkams turi priklausyti bent vienas žemės sklypas. Šį žymėjimą turime pakeisti į 0..*, kad įtrauktume galimybę, jog savininkai gali neturėti žemės sklypų.

Be to, jeigu žemės sklypai gali priklausyti keliems asmenims, mums reikėtų dar kartą pakeisti diagramą. Savininko žymėjimas (kairėje) nebegalėtų būti „1“. Vienetas nurodo, kad kiekvienas sklypas turi vieną ir tik vieną savininką, kas jau nebūtų tiesa. Sudėtingumą turime pakeisti į 1..*. Tai reiškia, kad sklypai vis dar turi priklausyti bent vienam savininkui, bet leidžiama daugiau negu vienas savininkas. Galime pastebėti, kad smulkus asociacijos sudėtingumo pakeitimas gali smarkiai įtakoti situacijas, kurios bus leidžiamos baigtoje duomenų bazėje.

Tikriausiai jau pastebėjote, kad abiem atvejais (skaitant iš kairės į dešinę ir iš dešinės į kairę) žymėjimai atskleidė mums du dalykus: mažiausią leistiną susijusių objektų skaičių ir didžiausią leistiną susijusio objektų skaičių. Tai vadinama **mažiausiu sudėtingumu** ir **didžiausiu sudėtingumu**. Mažiausias sudėtingumas yra kairėje taškų pusėje esantis skaičius, o didžiausias sudėtingumas yra dešinėje taškų pusėje esantis skaičius. 1-17 pav. parodyti mažiausi ir didžiausi sudėtingumai.



1-17 pav. Mažiausias ir didžiausias klasių diagramos sudėtingumas

Mažiausią sudėtingumą galima suprasti kaip privalomą/pasirinktinį žymėjimą. Mažiausias nulinis sudėtingumas reiškia, kad objektas *neprivalo turėti* ryšio su kita klase. Mažiausias 1 sudėtingumas reiškia, kad objektas *privalo* turėti ryšį bent su vienu kitos klasės elementu.

Pagal didžiausią sudėtingumą galima nustatyti bendrą ryšio kategoriją. Trys duomenų modelių ryšių tipai:

- **1:1** Vienas su vienu
- **1:M** Vienas su daugeliu – dar gali būti M:1, bet tai yra tas pats
- **N:M** Daugelis su daugeliu – dar gali būti žymima kaip M:M ar M:N

Kaip parodyta 1-17 pav. Mažiausias ir didžiausias klasių diagramos sudėtingumas , dvi didžiausios reikšmės yra 1 ir *, todėl šį ryšį suprantame kaip 1:M.

Šiuo bendru abstrakcijos lygiu sudėtingumams žymėti nenaudojame kitų verčių, išskyrus 0, 1 ir *. Studentai klausia, kaip galime apibūdinti situaciją, kai reikia vieną klasės elementą susieti su 1–5 kitos klasės elementais. Toks apribojimo tipas yra gana detalus. Mes jį sukursime įdiegę duomenų bazę ir taikydami *apribojimus* maksimaliam sudėtingumui (susijusių įrašų kiekiui). Šiame abstrakcijos lygyje mes priversti žymėti sudėtingumą kaip 1..*, o ne 1..5. Nepamirškite, kad kuriame loginį duomenų modelį ir bandome tik apibūdinti saugomos informacijos kategorijas ir kategorijų ryšius.

Išsamiai projektuodami duomenų modelį mes sukursime fizinį modelį, kuriame bus nurodyta, kaip baigtoje duomenų bazėje loginis modelis bus perkeliamas į lenteles, eilutes ir stulpelius. Fizinius modelius aptarsime kitame šio kurso skyriuje.

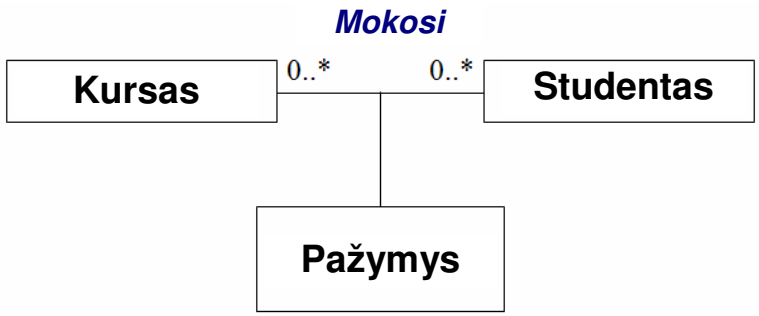
Asociacijų klasės

Kartais duomenų modelyje prie ryšio norime pridėti ypatybių ar atributų. Tokia situacija yra kai atributas turi būti naudojamas tik esant dviejų objektų asociacijai. Turime pridėti atributą prie ryšio, nes pridėjus atributą prie klasės sukeltume problemų.

Pavyzdžiui, įsivaizduokite ryšį tarp studento, studijuojamo kurso ir kurso pabaigoje gaunamo pažymio. Jeigu *pažymys* yra *kurso* objekto klasės atributas, kiekvienam elementui, pvz., GII-01 ar GII-05 priskiriamas vienas pažymys. Tai reiškia, kad tie, kurie pasirenka GII-05 gauna tokį patį pažymį.

Jeigu *pažymį* priskirtume *studentų* klasei, kiekvienam studentui būtų priskiriamas vienas pažymys. Tai reikštų, kad kiekvienas studentas už kiekvieną kursą gautų tokį patį pažymį.

Mums reikia sukurti situaciją, kai studentui besimokant kursą (arba susietam su juo) jam būtų priskirtas pažymys. Todėl sukuriame asociacijų klasę ir nurodome ją klasių diagramoje nubrėždami vertikalią liniją nuo asociacijos linijos ir pridėdami langelį, kuriame nurodome reikiamas ypatybes. Taip galime tiesiogiai pridėti *pažymį* prie asociacijos, o kiekvieną kartą studentui besimokant kursą jam priskiriamas naujas pažymys. 1-18 pav. parodyta, kaip atrodytų kurso ir studento ryšys.



1-18 pav. Asociacijų klasės pavyzdys

Geriausias būdas suprasti klasių diagramas yra peržiūrėti diagramas ir pabandyti jas sukurti patiems. Toliau apžvelgsime kelis pavyzdžius ir jums bus suteikta galimybė patiems kurti diagramas atliekant kurso dalies užduotis.

Diagramų pavyzdžiai:

Pirmame pavyzdyje pavaizduotas darbuotojo ir automobilio, kurį jam paskyrė darbdavys, pavyzdys. Gali būti, kad darbuotojai yra pardavėjai ir jiems reikia transporto priemonės, kad galėtų lankyti klientus.

Skaitydami diagramą iš kairės į dešinę galime suprasti, kad:

„*Darbuotojui skirtas 1 (ir tik 1) automobilis.*"

Nepamirškite, kad „1" yra „1..1" trumpinys, todėl mažiausia ir didžiausia vertė yra 1 automobilis. Mažiausias 1 sudėtingumas nurodo, kad tai yra privalomas ryšys ir darbdavys privalo skirti automobilį. Šiame modelyje negalime saugoti darbuotojų, kurie nedalyvauja pardavime, pvz., buhalterių ar administratorių, kuriems nėra priskirta transporto priemonių. Didžiausias sudėtingumas nurodo, kad darbuotojams negali būti priskirta daugiau negu viena transporto priemonė. Kiekvienam darbuotojui priskiriamas automobilis. Be išimčių.

Skaitydami diagramą iš dešinės į kairę galime suprasti, kad:

„Automobilis priskirtas 0 arba 1 darbuotojui."

Mažiausias nulinis sudėtingumas nurodo, kad tai yra pasirinktinis ryšys ir nėra būtina transporto priemonę priskirti darbuotojui (įmonei gali priklausyti niekam nepriskirti automobiliai). Didžiausias 1 sudėtingumas nurodo, automobilį galima priskirti tik vienam darbuotojui (t. y. automobilio negalima bendrai naudoti). Darome išvadą, kad įmonėje automobilių yra tiek pat, kiek darbuotojų, galbūt daugiau.

Skaitydami tik didžiausius sudėtingumus galime sakyti, kad ši asociacija yra ryšys vienas su vienu (1:1).

Antrame pavyzdyje parodytas bendrabučio pastatų ir juose gyvenančių studentų ryšys.

Skaitydami diagramą iš kairės į dešinę suprantame, kad:

„Bendrabutyje gyvena nulis arba daugiau studentų.“

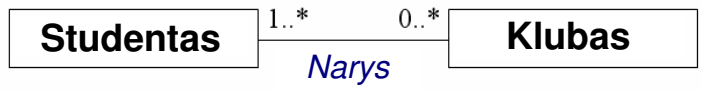
Mažiausias nulinis sudėtingumas nurodo, kad bendrabučiuose nebūtinai privalo gyventi studentai (t. y. bendrabučiai gali būti tušti). Didžiausias sudėtingumas nurodo, kad tam tikrame bendrabutyje gali gyventi bet koks studentų skaičius. Kaip jau aptarėme, loginiame modelyje nenustatomas tikrasis didžiausias pastate galinčių gyventi studentų skaičius. Gali būti, kad jame vienu metu gali gyventi daugiausiai 100 studentų. Tikrieji studentų skaičiaus apribojimai bus taikomi kuriant duomenų bazę ir nurodant duomenų bazės apribojimus. Savo tikslais kuriame tik bendras diagramas ir bet kuris didžiausias sudėtingumas > 1 bus nurodytas * žymėjimu.

Skaitydami diagramą iš dešinės į kairę suprantame, kad:

„Studentas gyvena 0 arba 1 bendrabutyje.“

Ir vėl, mažiausias nulinis sudėtingumas nurodo, kad studentai neprivalo gyventi bendrabutyje. Daug studentų pasirenka gyventi nuomojamuose butuose ar už universiteto teritorijos ribų esančiuose pastatuose. Didžiausias 1 sudėtingumas nurodo, kad jeigu studentas gyvena bendrabutyje, jis gali gyventi tik viename bendrabutyje (t. y. studentai negali būti priskirti dviems kambariams).

Skaitydami didžiausius sudėtingumus galime tvirtinti, kad bendrabučio-studento asociacija yra vieno su daugeliu asociacija (1:M).



Trečiame pavyzdyje pavaizduotas studentų ir klubų ryšys. Klubai gali būti universiteto sporto ar hobio organizacijos, pvz., plaukimo klubas ar fotografavimo klubas, kuriose renkasi panašius pomėgius turintys studentai.

Skaitydami iš kairės į dešinę suprantame, kad:

„Studentas gali būti 0 ar daugiau klubų narys.“

Minimalus nulinis sudėtingumas nurodo, kad tai yra pasirinktinė asociacija ir studentai neprivalo įstoti į klubą, Didžiausia * reikšmė nurodo, kad studentas gali įstoti į tiek klubų, kiek panorės.

Skaitydami iš dešinės į kairę suprantame, kad:

„Klubui gali priklausyti vienas ar daugiau studentų narių.“

Mažiausia 1 reikšmė nurodo, kad tai privaloma asociacija ir klubui, tam, kad jis egzistuotų, turi priklausyti bent vienas studentas. Didžiausia * reikšmė nurodo, kad nėra apribojimų, kiek studentų gali įstoti į tam tikrą klubą.

Skaitydami didžiausius sudėtingumus šią asociaciją apibūdintume kaip daugelis su daugeliu asociacija (M:M).

Galite pastebėti, kad labai paprasta diagrama perteikia nemažai informacijos apie duomenų modelyje naudojamus ryšius ir situacijų tipus. Labai nedideli modelio pakeitimai gali smarkiai pakeisti situacijas, kurias gali apdoroti mūsų duomenų modelis. Pavyzdžiui, 1-19 pav. asociacija parodyta bendrabučio studento asociacija, kurią matėme antrame pavyzdyje, išskyrus tai, kad kairysis sudėtingumas pakeistas iš 0..1 į 1. Dabar mūsų modelis negalės apdoroti studentų, kurie gyvena už universiteto teritorijos ribų. Pagal šį modelį visi studentai privalo gyventi universiteto teritorijoje.

1-19 pav. Modifikuota bendrabučio studento asociacija

Kurdami duomenų modelius dažnai turėsite kurti duomenų modelį pagal pokalbius ar rašytinius apibūdinimus, kuriuose nurodoma, kas turi būti saugoma duomenų bazės sistemoje ir kokias funkcijas ji turi atlikti. Turėsite nustatyti asociacijas ir sudėtingumus atsižvelgdami į verslo taisykles arba organizacijai taikomus apribojimus ir nurodyti, kaip turi būti įgyvendinami verslo tikslai. Pavyzdžiui, turėsite išsiaiškinti, ar studentai gali gyventi už universiteto teritorijos ribų, ar jie visi turi gyventi bendrabutyje.

Šio kurso užduočių ar egzaminų metu jums dažnai nebus pateikiamas išsamus duomenų modelio verslo taisyklių apibūdinimas. Tada, kurdami savo modelį, turėsite pasitikėti savo žiniomis apie ryšius. Pagrindinė šio proceso dalis yra savo prielaidų išdėstymas, kad užduotį ar egzamino atsakymus vertinantis asmuo suprastų, kaip jūs įsivaizduojate ryšį. Nepamirškite: gali nebūti nė vieno teisingo atsakymo! Jeigu būsite paprašyti sukurti mūsų aptartą bendrabučio studento asociacija, bet nebus aiškiai nurodyta, ar studentai gali gyventi už universiteto teritorijos ribų arba, kad mes saugome tik bendrabutyje gyvenančius studentus, turėsite pasirinkti vieną pritaikymą ir pagrįsti savo pasirinkimą arba

savo prielaidas, kuriomis remdamiesi pasirikote konkretų variantą. Galite sukurti panašų modelį:

> *„Darau prielaidą, kad šios duomenų bazės tikslas – valdyti bendrabučius, galbūt tvarkyti naujų studentų registravimus ir bendrabutyje gyvenančių studentų atsiskaitymus ir t. t. Nėra prasmės saugoti informacijos apie studentus, kurie negyvena universiteto teritorijoje, nes jie nedaro įtakos mūsų tikslams.“*

Tada turėsite sukurti diagramą, panašią į 1-19 pav. ir po ja parašyti savo prielaidas. Jūsų prielaidos gali būti visai paprastos:

Prielaidos:

- *Duomenų bazės tikslas – valdyti bendrabučius, todėl visi šiam modeliui priklausantys studentai privalo gyventi bendrabutyje.*
- *Turi būti galimybė palikti tuščius bendrabučius, nes vasarą juose gyvena labai mažai studentų.*

Kitame pavyzdyje sukurkime klasių diagramą, skirtą dviejų klasių asociacijai: konsultantas ir studentas. Čia apibūdinama universitetinio konsultavimo situacija, kai konsultantas padeda studentams kurį kursą rinktis ar kokio mokslinio laipsnio siekti. Jei nėra jokios papildomos informacijos, mes nežinome, kaip veikia ši asociacija. Pavyzdžiui: ar studentas privalo konsultuotis su konsultantu? Ar studentas paprasčiausiai užeina ir pasitaria su galimu konsultantu, ar studentai turi susitarti dėl susitikimų ir susitikti su tam tikru konsultantu, kuris išmano padėtį? Mes turime sukurti šios situacijos prielaidas ir sugalvoti jos funkcijas. Taip jūsų darbą vertinantis asmuo galės suprasti, kodėl jūsų atsakymas skiriasi nuo kito studento atsakymo.

Jūsų atsakymas dėl konsultanto studento asociacijos galėtų atrodyti šitaip:

Prielaidos:

- *Bet kuris konsultantas gali patarti keliems studentams, bet privalo patarti bent vienam. Jeigu jis niekam nepataria, jis nėra konsultantas.*
- *Studentai neprivalo naudotis konsultanto paslaugomis, bet, jeigu naudojasi, gali kreiptis į bet kurį galimą konsultantą.*

Praktinės užduotys

Pabandykite sudaryti šias klasių diagramas. Surašykite visas prielaidas, kurios padėjo jums apsispręsti. Atsakymai surašyti po užduotimis esančiame skyriuje.

1. Sukurkite modelį panaudodami paveikslo ir galerijos klases. Bandome pavaizduoti situaciją, kai galerijoje eksponuojami (rodomi) paveikslai.
2. Sukurkite modelį, panaudodami pastato ir buto klases ir asociaciją, nurodančią, kuriuose pastatuose fiziškai yra tam tikri butai.
3. Sumodeliuokite klases knyga, leidėjas ir autorius. Gali ir nebūti visų klasių asociacijų (kaip jums atrodo tinkamiau).
4. Sumodeliuokite klases transporto priemonė, automobilis ir motociklas. Patarimas: šis modelis yra neįprastas.
5. Sukurkite universiteto studijų skyriaus klasių diagramą. Studijų skyrius valdo kiekvieno kurso duomenis, įskaitant informaciją apie kurso dėstytoją, paskaitų laiką, vietą ir kiekvienos grupės studentus, jų vardus, pavardes, numerius ir gyvenamuosius bendrabučius. Studentui užsiregistravus klausyti kursą įrašomas jo pažymys.
6. Sudarykite bibliotekos klasių diagramą. Reikia rinkti informaciją apie visus išduotus leidinius (toliau elementus), įskaitant vaizdo įrašus ir knygas. Be to, reikia tvarkyti studentų ir elementų, kuriuos jie gali pasiskolinti, informaciją, įskaitant elementų paėmimo ir grąžinimo laikus. Pridėkite susijusių atributų. Nurodykite reikiamas prielaidas.

Sprendimai:

Bus daug sprendimų variantų, atsižvelgiant į studento modeliuojamos situacijos suvokimą. Svarbu, kad studentų žymėjimai (pvz., sudėtingumas) atitiktų jų daromas prielaidas.

Studentai kartais pasimeta supratę, kad yra ne vienas teisingas atsakymas, ypač kai neturime visos išsamios informacijos. Patariame prieš projektuojant nepasitikėti vien tik savo žiniomis ir pasitarti su tuo, kas išmano šią verslo sritį. Šio kurso ir egzamino metu mes neturėsime išsamių šios srities žinių. Tada jums reikės NURODYTI SAVO PRIELAIDAS.

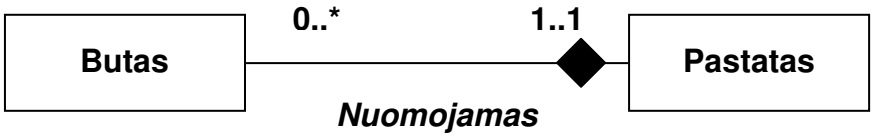
Toliau nurodyti labiausiai tikėtini sprendimai, bet gali būti ir kitų pagrįstų atsakymų.

1. **Sukurkite modelį panaudodami paveikslo ir galerijos klases. Bandome pavaizduoti situaciją, kai galerijoje eksponuojami (rodomi) paveikslai.**

Prielaidos:
- Kuriant šį modelį reikia atsižvelgti į laiką – paveikslas negali būti eksponuojamas dviejose galerijose tuo pačiu metu.
- Nebūtina eksponuoti paveikslų.
- Galerijoje gali būti eksponuojamos skulptūros ar kiti vaizduojamosios dailės projektai, todėl nėra būtina eksponuoti paveikslus.

Pastabos:
- Sudėtingumai gali būti 1..* kairėje pusėje ir 0..*, 1..1 arba 1..* dešinėje, atsižvelgiant į studentų prielaidas.

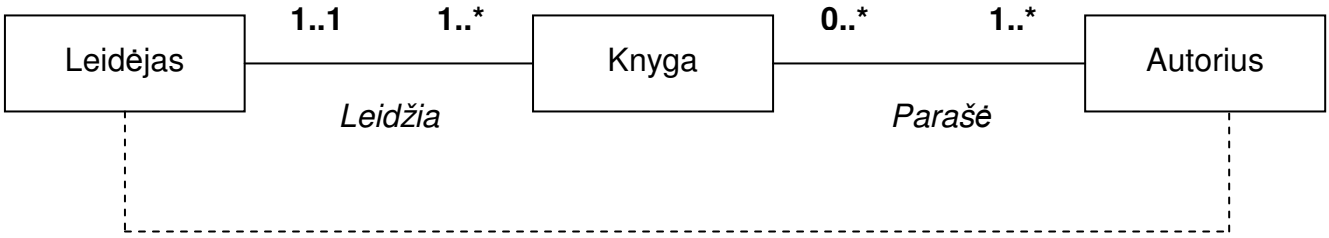
2. **Sukurkite modelį, panaudodami pastato ir buto klases ir asociaciją, nurodančią, kuriuose pastatuose fiziškai yra tam tikri butai.**



Prielaidos:
- Pastatas gali būti negyvenamasis, todėl jame butų gali ir nebūti.

Pastabos:
- Šiame modelyje nedaug kintamųjų dydžių. Kairėje pusėje studentai galėtų įvesti 1..* ir nurodyti, kad tvarkomi tik gyvenamieji pastatai, kuriuose yra butų, todėl kiekviename pastate turi būti bent vienas butas.
- Tai pakankamai aiškus komponavimo asociacijos pavyzdys, panašiai kaip per paskaitas pateiktas pavyzdys su bendrabučio kambariais.

3. **Sumodeliuokite klases knyga, leidėjas ir autorius. Gali ir nebūti visų klasių asociacijų (kaip jums atrodo tinkamiau).**

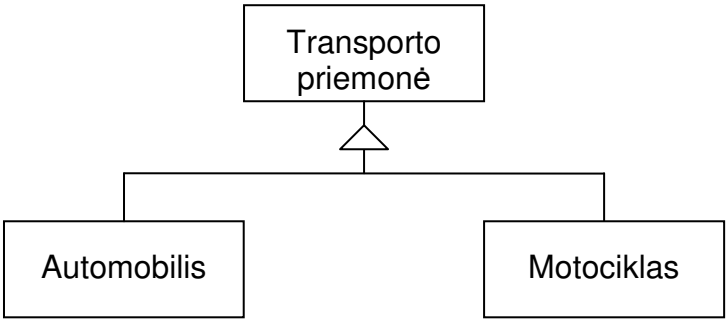


Ar čia yra ryšys?

Prielaidos:
- Knygą gali išleisti leidėjas, bet ji turi būti išleista, kad galėtume vadinti ją knyga
- Leidykla turi išleisti bent vieną knygą
- Knygos autoriai gali būti keli (pvz., vadovėlis)
- Autoriaus neprivalo parašyti knygos (pvz., poetai ir t. t.)

Pastabos:
- Atsižvelgiant į tai, ar studentai mano, kad knygą gali išleisti keli leidėjai ir t. t., šiame modelyje gali būti nemažai kintamųjų. Reikia turėti omenyje studentų prielaidas ir pavyzdžius.
- Įdomiausia yra nustatyti, ar yra ryšys tarp autoriaus ir leidėjo, išskyrus tą faktą, kad knygą parašo autorius, o išleidžia tam tikras leidėjas. Jei tai yra vienintelis ryšys, jis kuriamas naudojant knygų klasę. Kai kurie studentai pasiūlė, kad autorius gali turėti ryšį su leidėju dar prieš pradėdamas rašyti knygą. Gali būti, kad jam iš anksto sumokėjo už knygos parašymą.

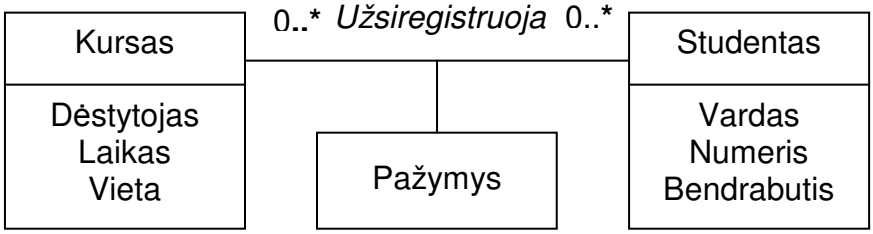
4. **Sumodeliuokite klases transporto priemonė, automobilis ir motociklas. Patarimas: šis modelis yra neįprastas.**



Pastabos:
- Tai pakankamai aiškus paveldimumo asociacijos pavyzdys. Automobiliai ir motociklai yra transporto priemonių rūšys – automobilis „yra“ transporto priemonė.
- Atkreipkite dėmesį, kad naudodami paveldimumo asociacijas sudėtingumo nenaudojame.

5. **Sukurkite universiteto studijų skyriaus klasių diagramą. Studijų skyrius valdo kiekvieno kurso duomenis, įskaitant informaciją apie kurso dėstytoją,**

paskaitų laiką, vietą ir kiekvienos grupės studentus, jų vardus, pavardes, numerius ir gyvenamuosius bendrabučius. Studentui užsiregistravus klausyti kursą įrašomas jo pažymys.



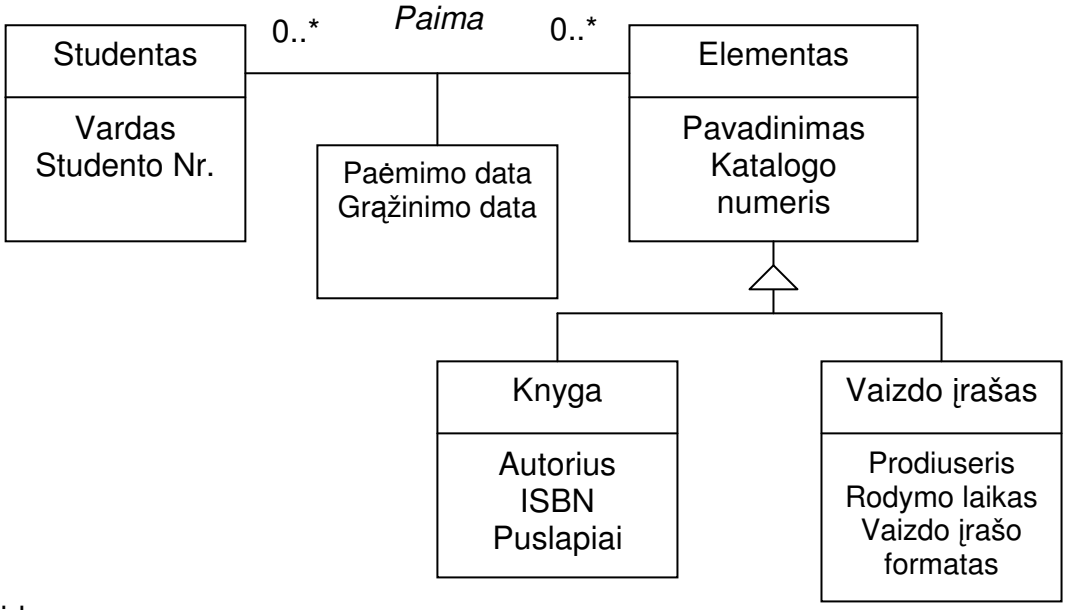
Prielaidos:

- Studentai nebūtinai turi užsiregistruoti klausyti kursą (gali būti, kad kursai nėra siūlomi)
- Studentai neprivalo užsiregistruoti, kad klausytų kursą (jie gali lankyti kursą, tada jie negauna pažymių, bet privalo mokėti mokestį ir t. t.)

Pastabos:

- Šiame pavyzdyje nėra aiškiai pateiktų klasių, todėl jas reikia nustatyti pagal tekstą.
- Dar turime nurodyti ypatybes. Ne tik klasės pavadinimą.
- Pažymys turi būti asociacijos klasė, nes, kaip aptarėme per paskaitą, pažymys turi būti taikomas atsižvelgiant į pateikto kurso asociaciją, kad tam tikras studentas būtų veiksmingas.
- Čia susiduriame su tokiais daiktavardžiais kaip bendrabutis ir dėstytojas. Projektuodami turime nuspręsti, ar šie daiktavardžiai yra klasės, ar tik kitos klasės atributai. Pavyzdžiui, dėstytojas gali sudaryti atskirą klasę, bet gali būti ir klasės kursas atributu. Pagrindinis klausimas yra 1) ar dar ką nors žinome apie dėstytoją (darbo valandas, mokslinių tyrimų temas ir t. t.), ar ne. Jeigu kitų atributų, išskyrus vardą, neturime gal reikėtų skirti dėstytoją kurso atributu. 2) ar studijų skyriuje turi būti tvarkoma informacija apie dėstytoją. Atsižvelgiant į šių duomenų vartotoją reikia nuspręsti, kas yra klasė ir kas yra tiktai atributas.
- Bendrabutyje susidursime su tokia pačia situacija – jeigu netvarkome informacijos apie bendrabučio kambarius ir žinome tik pastato numerį, gali būti, kad bendrabutis yra studento atributas, o ne atskira klasė.
- Bendrabutį ir dėstytoją sumodeliavau kaip ypatybes, bet studentas gali būti pakankamas argumentas šioje diagramoje išskirti juos į dvi atskiras klases.
- Kai kurie studentai klausė, ar studijų skyrius yra atskira klasė. Jeigu kuriame vieno universiteto modelį, tada jame yra tik vienas studijų skyrius ir nėra būtina išsaugoti šios informacijos. Jei modeliuojame kelių universitetų paslaugas, tada reikia išsaugoti studijų skyrius, kad galėtume nurodyti konkrečių universitetų siūlomus kursus.

6. **Sudarykite bibliotekos klasių diagramą. Reikia rinkti informaciją apie visus išduotus elementus, įskaitant vaizdo įrašus ir knygas. Be to, reikia tvarkyti studentų ir elementų, kuriuos jie gali pasiskolinti, informaciją, įskaitant elementų paėmimo ir grąžinimo laikus. Pridėkite susijusių atributų. Nurodykite reikiamas prielaidas.**



Prielaidos:

- Elementų paimti nebūtina, bet juos gali paimti keli studentai
- Studentai neprivalo ką nors paimti iš bibliotekos, bet gali paimti kelis elementus

Pastabos:

- Čia neturėtų būti daug atsakymo variantų. Studentai gali atvaizduoti knygos/vaizdo įrašo sąvoką naudodami elemento atributą, pavadintą elemento tipas ar kaip nors panašiai. Tai būtų priimtina.
- Jeigu norime saugoti paėmimų retrospektyvą, elemento paėmimo ir grąžinimo datos turi būti asociacijų klasė, kad kiekvieną kartą studentui paėmus elementą galėtume išsaugoti paėmimo ir grąžinimo datas.
- Kai kurie studentai klausė, ar biblioteka yra klasė. Biblioteka yra viena, kaip ir studijų skyrius, be to, joje atliekamos visos operacijos, todėl šios informacijos išsaugoti nereikia.

Modeliavimo patarimai

Nepamirškite: modeliai skirti bendravimui. Pagrindinė klasių diagramos paskirtis – aptarti ir dokumentuoti duomenų modelį, todėl aiškumas turi būti viena iš pagrindinių aplinkybių. Modeliai gali būti labai sudėtingi. Tačiau paprastas ir duomenų reikalavimus atitinkantis modelis bus efektyvesnis ir produktyvesnis.

Nepamirškite: įtraukite tik svarbius dalykus. Duomenų modelio kūrimas yra kaip tvoros statymo užduotis. Projektuotojas turi nuspręsti, kuri informacija yra svarbi ir kuri – ne. Nereikia įtraukti į modelį nesusijusių duomenų. Taip pagreitinsite baigtos sistemos duomenų apdorojimą ir sumažinsite informacijos saugojimo vietos dydį. Dažnai sprendimas, *ką modeliuoti,* yra toks pat svarbus kaip sprendimas, kaip modeliuoti.

Klasių ir asociacijų nustatymas. Kartais pagal tekstą sudėtinga nuspręsti, ką reikėtų priskirti klasei ir ką asociacijai.

- Klasės dažniausiai apibūdinamos daiktavardžiais.
- Asociacijos dažniausiai apibūdinamos veiksmažodžiais.

Klasių ir ypatybių nustatymas. Kartais sunku atskirti ypatybes nuo klasių, nes jos dažniausiai apibūdinamos daiktavardžiais. Pavyzdžiui, aptariant 5 pavyzdį (universiteto studijų skyriaus duomenų bazė) studentai dažnai klausia, ar *dėstytojas* yra kurso ypatybė, ar klasė, turinti asociaciją su kursu.

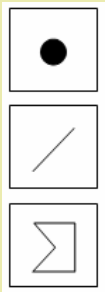
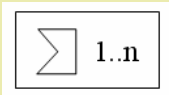
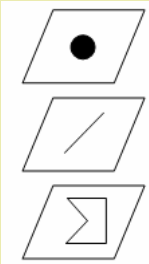
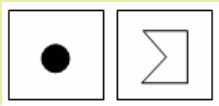
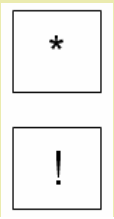
- Jeigu kursą dėsto tik vienas dėstytojas ir mes apie jį daugiau nieko nežinome, tikriausiai geriausia jį modeliuoti kaip kurso ypatybę.
- Jeigu turime daugiau žinių apie dėstytoją (dėstytojų kambario vietą ar atliekamų tyrimų specializaciją) arba jeigu kursą dėsto daugiau negu vienas dėstytojas, tikriausiai reikėtų sukurti naują klasę ir joje išsaugoti šią informaciją.
- Ši problema susijusi su ryšiais tarp vienos klasės atributų ir ryšiais tarp skirtingų klasių. Ryšiai tarp vienos klasės atributų vadinami funkcine priklausomybe. Mes išsamiai ją aptarsime nagrinėdami kitos paskaitos temą.

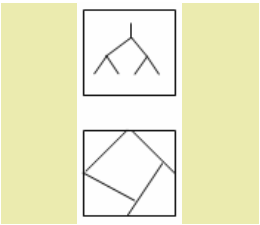
Erdvinių duomenų modeliavimas

Standartinė klasių diagrama dažniausiai atitinka daugumą duomenų bazės projektavimo reikalavimų. Tačiau atvaizduojant erdvinius elementus ir erdvinius ryšius ji turi trūkumų. Pastaruoju metu buvo pasiūlyta (pvz., Shekhar, 2003 m.) standartinį būsenų ryšio (Entity Relationship) ar UML klasių diagramos žymėjimą papildyti šiais erdviniais elementais. **Piktogramos** dažnai pridedamos prie žymėjimo. Jos nurodo erdvinę modelio semantiką.

Toliau esančioje 1-4 lentelėje pateikta piktogramų, kurios gali būti naudojamos klasių diagramoje, suvestinė.

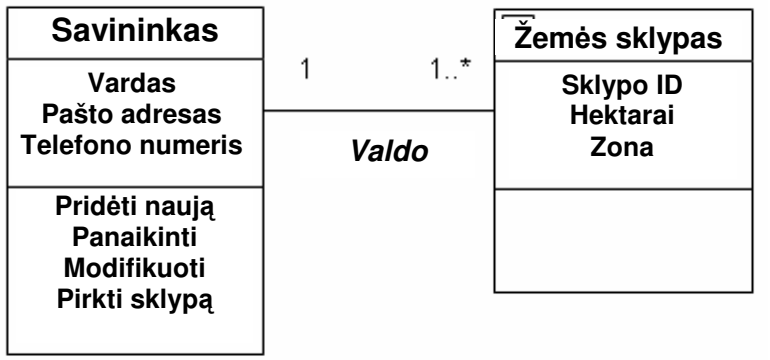
1-4 lentelė Piktogramų elementų suvestinė

Piktograma	Reikšmė
Pagrindinės formos 	Vektoriniame modelyje geoobjektai duomenų bazėje atvaizduojami kaip taškai, linijos ar poligonai. Gali būti naudojamos šie pagrindiniai formų tipai: ▪ Taškinis geoobjektas ▪ Linijinis geoobjektas ▪ Plotinis geoobjektas (poligonas)
Sudėtiniai geoobjektai  **1.2.2.1**	Tam tikri geoobjektai atvaizduojami grupuojant paprastas formas. Pavyzdžiui, atvaizduojant Indonezijos salas keli poligonai turės būti laikomi vienu geoobjektu. Tokiu atveju galėsime atvaizduoti pagrindinių formų rinkinio sudėtingumą kaip žymėjimo dalį. Sudėtingumus galima atvaizduoti taip, kaip jie vaizduojami klasių diagramoje: ▪ 0..1 ▪ 1..1 ▪ 0..n ▪ 1..n ▪ n..m
Išvestinė forma 	Kai kuriuos geoobjektus GIS sistemoje sudaro geoobjektų grupės. Pavyzdžiui, erdvinis Europos atvaizdavimas yra ne kas kita kaip atskirų poligonų, vaizduojančių šalis nares, grupavimas. Tada formą atvaizduojame pasviruoju simboliu.
Papildomas atvaizdavimas 	Kai kuriuos geoobjektus galima atvaizduoti naudojant skirtingų tipų formas, atsižvelgiant į atvaizdavimo mastelį. Pavyzdžiui, naudojant labai smulkų mastelį miestas gali būti atvaizduotas kaip taškas, bet padidinus mastelį jis gali būti atvaizduotas kaip poligonas. Tada gali būti nurodytos kelių tipų formos.
Vartotojo nurodyta forma 	Kartais kyla neįprastų situacijų, kai naudojant vieną formą negalima atvaizduoti erdvinės reikšmės. Pavyzdžiui, miesto vandentiekio tinklas gali būti sudarytas iš įvairių formų tipų: sklendės kaip taškai, vamzdžiai kaip linijos ir talpos kaip poligonai. Tada bus naudojamas žymėjimas *. Dar neįprastesnėms sąlygoms vaizduoti gali būti naudojamas simbolis „!“ ir jo paaiškinimas.
Asociacijų žymėjimai	Norėdami parodyti erdvinę reikšmę, galime pridėti žymėjimų prie asociacijų: ▪ Dalis – tinklas: geoobjektai sudaro linijų tinklo dalį.



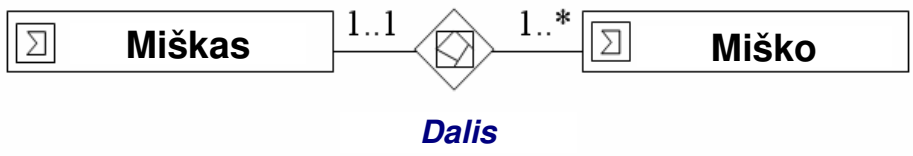
- Dalis – padalijimas: geoobjektai sudaro poligonų erdvės skaidymo dalį, pvz., žemėnaudos poligonai.

Pavyzdžiui, paprastoje pirmiau pavaizduotoje savininko–žemės sklypo diagramoje prie žemės klypo klasės galime pridėti piktogramą ir nurodyti, kad sklypai yra plotiniai geoobjektai. Prie savininko klasės piktogramos nepridedamos, nes savininkas nėra erdvinis objektas. Rezultatas gali atrodyti kaip 1-20 pav. pavaizduota diagrama su piktograma.



1-20 pav. Savininko-žemės sklypo diagrama su piktograma

1-21 pav. parodytas piktogramos, žyminčios asociaciją, naudojimo pavyzdys.



1-21 pav. Miško-miško medyno asociacija

Reliacinis modelis

Šioje temoje išsamiau panagrinėsime labiausiai paplitusią duomenų bazių struktūrą – reliacinį modelį. Reliacinės duomenų bazės sudaro daugumą pasaulyje įdiegtų duomenų bazių, ir jų koncepcija ypač svarbi norint suprasti GIS duomenų struktūras ir panaudojimą.

Šioje temoje pateikta teoriniu požiūriu daugiausia pastangų pareikalausianti kurso medžiaga. Ji pakankamai abstrakti, o likusi kurso dalis yra daugiausia techninė ir pritaikoma realybėje. Kartu su šios paskaitos medžiaga pateikiama daug pavyzdžių, o savarankiškai atliekami pratimai pateikiami su jų sprendimais, taigi, studentai gali praktiškai išmėginti šią medžiagą.

1.2.3 Terminologija

Nagrinėdami duomenų bazes susidursite su daugeliu terminų, kurie reiškia tą patį dalyką. Tolesnėse iliustracijose išnagrinėsime tris sąvokas, kaip jas gali vartoti skirtingoms bendruomenėms priklausantys su GIS dirbantys žmonės.

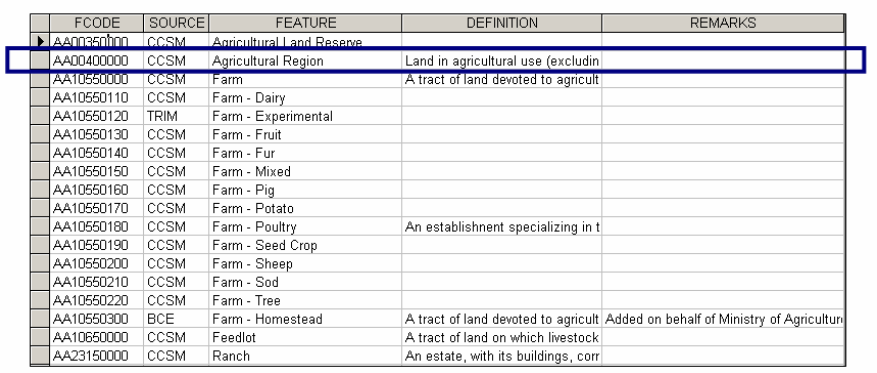
- **Naudotojas**, tas, kas naudoja programinę įrangą ir turi bendrą supratimą apie jos pagrindą sudarančias struktūras.
- **Reliacinė algebra** yra matematikos ir kompiuterijos sritis, nagrinėjanti tokius dalykus, kaip aibių teorija.
- **Programuotojo** požiūris į duomenų struktūras yra praktiškesnis.

Atlikdami užduotis ir laikydami egzaminus galite laisvai vartoti bet kuriuos iš šiose iliustracijose surašytų terminų.

FCODE	SOURCE	FEATURE	DEFINITION	REMARKS
AA00350000	CCSM	Agricultural Land Reserve		
AA00400000	CCSM	Agricultural Region	Land in agricultural use (excludin	
AA10550000	CCSM	Farm	A tract of land devoted to agricult	
AA10550110	CCSM	Farm - Dairy		
AA10550120	TRIM	Farm - Experimental		
AA10550130	CCSM	Farm - Fruit		
AA10550140	CCSM	Farm - Fur		
AA10550150	CCSM	Farm - Mixed		
AA10550160	CCSM	Farm - Pig		
AA10550170	CCSM	Farm - Potato		
AA10550180	CCSM	Farm - Poultry	An establishnent specializing in t	
AA10550190	CCSM	Farm - Seed Crop		
AA10550200	CCSM	Farm - Sheep		
AA10550210	CCSM	Farm - Sod		
AA10550220	CCSM	Farm - Tree		
AA10550300	BCE	Farm - Homestead	A tract of land devoted to agricult	Added on behalf of Ministry of Agricultur
AA10650000	CCSM	Feedlot	A tract of land on which livestock	
AA23150000	CCSM	Ranch	An estate, with its buildings, corr	

Naudotojas: Lentelė
Reliacinė algebra: Ryšys
Programuotojas: Failas

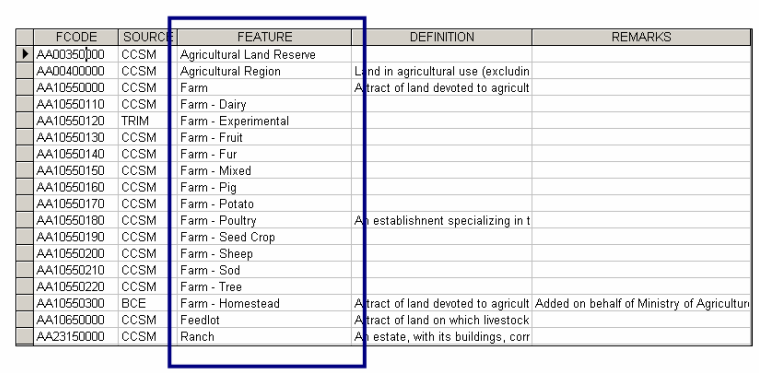
1-22 pav. Visa lentelė



FCODE	SOURCE	FEATURE	DEFINITION	REMARKS
AA00350000	CCSM	Agricultural Land Reserve		
AA00400000	CCSM	Agricultural Region	Land in agricultural use (excludin	
AA10550000	CCSM	Farm	A tract of land devoted to agricult	
AA10550110	CCSM	Farm - Dairy		
AA10550120	TRIM	Farm - Experimental		
AA10550130	CCSM	Farm - Fruit		
AA10550140	CCSM	Farm - Fur		
AA10550150	CCSM	Farm - Mixed		
AA10550160	CCSM	Farm - Pig		
AA10550170	CCSM	Farm - Potato		
AA10550180	CCSM	Farm - Poultry	An establishnent specializing in t	
AA10550190	CCSM	Farm - Seed Crop		
AA10550200	CCSM	Farm - Sheep		
AA10550210	CCSM	Farm - Sod		
AA10550220	CCSM	Farm - Tree		
AA10550300	BCE	Farm - Homestead	A tract of land devoted to agricult	Added on behalf of Ministry of Agricultur
AA10650000	CCSM	Feedlot	A tract of land on which livestock	
AA23150000	CCSM	Ranch	An estate, with its buildings, corr	

Naudotojas: Eilutė
Reliacinė algebra: Eilė
Įrašas
Programuotojas:

1-23 pav. Viena lentelės eilutė



FCODE	SOURCE	FEATURE	DEFINITION	REMARKS
AA00350000	CCSM	Agricultural Land Reserve		
AA00400000	CCSM	Agricultural Region	Land in agricultural use (excludin	
AA10550000	CCSM	Farm	A tract of land devoted to agricult	
AA10550110	CCSM	Farm - Dairy		
AA10550120	TRIM	Farm - Experimental		
AA10550130	CCSM	Farm - Fruit		
AA10550140	CCSM	Farm - Fur		
AA10550150	CCSM	Farm - Mixed		
AA10550160	CCSM	Farm - Pig		
AA10550170	CCSM	Farm - Potato		
AA10550180	CCSM	Farm - Poultry	An establishnent specializing in t	
AA10550190	CCSM	Farm - Seed Crop		
AA10550200	CCSM	Farm - Sheep		
AA10550210	CCSM	Farm - Sod		
AA10550220	CCSM	Farm - Tree		
AA10550300	BCE	Farm - Homestead	A tract of land devoted to agricult	Added on behalf of Ministry of Agricultur
AA10650000	CCSM	Feedlot	A tract of land on which livestock	
AA23150000	CCSM	Ranch	An estate, with its buildings, corr	

Naudotojas: Stulpelis
Reliacinė algebra: Atributas
Laukas
Programuotojas:

1-24 pav. Vienas lentelės stulpelis

1.2.4 Ryšiai

Išsiaiškinome, kad reliacinio modelio pagrindas yra ryšys, ir kad ryšys labai panašus į lentelę. Kad lentelę būtų galima laikyti ryšiu, formaliu požiūriu ji turi atitikti tokius penkis kriterijus:

1. Visi tam tikro stulpelio įrašai turi būti to paties tipo. „To paties tipo“ reiškia, kad visi atitinkamo stulpelio įrašai yra skaičiai, arba visi tokie įrašai yra datos. Nenorėsime maišyti tekstinių duomenų su skaičiais ir datomis.

2. Kiekvienas stulpelis turi nesikartojantį pavadinimą. Kiekvienam stulpeliui suteikiamas pavadinimas, kuris aprašo, kokie duomenys saugomi tame stulpelyje, ir šie pavadinimai lentelėje turi būti unikalūs (t. y., toks pat pavadinimas gali būti naudojamas *skirtinguose* ryšiuose, tačiau negali būti dukart *toje pačioje* lentelėje).

3. Lentelėje negali būti identiškų eilučių. Jei dviejose eilutėse visų stulpelių informacija yra visiškai tokia pat, ši lentelė nėra ryšys.

4. Stulpelių tvarka nėra svarbi. Nepageidautina, kad stulpelių tvarka turėtų kokią nors numanomą reikšmę. Stulpelius turėtų būti galima kaitalioti vietomis, ir tai neturėtų įtakos duomenų prasmei.

5. Eilučių tvarka nėra svarbi. Tas pat, kaip minėta 4 punkte, bet šį kartą nepageidaujama, kad kokią nors numanomą reikšmę turėtų eilučių tvarka. Jei turite lentelę *Penkios svarbiausios priežastys mokytis GIS*, ji gali netenkinti mūsų testo, nes tai, kad eilutė yra pirma, gali reikšti, kad tai yra svarbiausia priežastis. Mes galime įgyvendinti tą patį dalyką nepažeisdami 5 punkto sąlygos, jei, pavyzdžiui, pridėsime stulpelį, pavadintą *Svarba*, ir įrašysime skaičius nuo 1 iki 5, kad pažymėtume, kokia svarbi yra priežastis. Ryšys turi turėti galimybę keisti eilučių tvarką nekeičiant duomenų prasmės.

Kaip principo iliustraciją išnagrinėkite toliau pateiktas lenteles ir pagalvokite, ar jos tenkina penkis prieš tai išdėstytus kriterijus. Nustatykite kriterijus, kurių lentelė netenkina.

1-5 lentelė. Pirmas lentelės pavyzdys

Prekės nr.	Aprašymas	Aprašymas	Kaina
10035	Sėdynės užvalkalas	Dėklas	50 JAV dol.
26658	Oro gaiviklis	Pušis	3 JAV dol.
6685	Padanga	14R42	80 JAV dol.
35891	Domkratas	2 tonų	95 JAV dol.

1-5 lentelė turi du stulpelius, pavadintus „Aprašymas“. Nepaisant to, kad juose iš tiesų saugoma šiek tiek skirtinga informacija, dėl tokio pasikartojančio stulpelių pavadinimo ši lentelė netenkina 2 punkto testo, ir todėl nėra ryšys.

1-6 lentelė. Antras lentelės pavyzdys

Prekės nr.	Pavadinimas	Aprašymas	Pastaba
10035	Sėdynės užvalkalas	Dėklas	50 JAV dol.
26658	Oro gaiviklis	Pušis	Taip
6685	Padanga	14R42	80 JAV dol.
35891	Domkratas	2 tonų	2006 m. sausio 1 d.

1-6 lentelė dabar turi unikalius stulpelių pavadinimus, be to, buvo pakeistas paskutinis stulpelis. Šiame stulpelyje dabar yra įvairiausi skirtingų tipų duomenys: valiuta, dvejetainiai (taip, ne) ir data. Dėl to lentelė netenkina 1 punkto testo, ir todėl nėra ryšys.

1-7 lentelė. Trečias lentelės pavyzdys

Prekės nr.	Aprašymas	Kaina
10035	Sėdynės užvalkalas	50 JAV dol.
26658	Oro gaiviklis	3 JAV dol.
6685	Padanga	80 JAv dol.
10035	Sėdynės užvalkalas	50 JAV dol.

1-7 lentelė turi pasikartojančių eilučių (pirma ir paskutinė eilutės yra tokios pat visiems stulpeliams). Ši lentelė pažeidžia 3 punkto tekstą, taigi, nėra ryšys.

1.2.5 Funkcinė priklausomybė

Dažnai pasitaiko, kad du tos pačios lentelės stulpeliai vienas su kitu turi tam tikrą ryšį. Pavyzdžiui, lentelėje, kurioje skirtinguose stulpeliuose, bet toje pačioje eilutėje yra jūsų vairuotojo teisių numeris ir jūsų pavardė, tarp šių dviejų informacijos elementų yra aiškus ryšys.

Funkcinė priklausomybė yra ryšys tarp to paties ryšio atributų. Pateikę vieno atributo reikšmę galite nustatyti, apskaičiuoti arba surasti kito atributo reikšmę.

Pavyzdžiui, jei didelėje kredito kortelių balanso lentelėje žinome kliento sąskaitos Nr. reikšmę, galime surasti kliento sąskaitos balansą, ir būti tikri, kad peržiūrėjome reikiamą balansą. Jei tai tiesa, kliento sąskaitos balansas **funkciškai priklauso** nuo kliento

sąskaitos Nr. Arba dar galime sakyti, kad kliento sąskaitos Nr. **apibrėžia** kliento sąskaitos balansą.

Formalesnis apibrėžimas būtų:

> **Turint R ryšį, ryšio atributas Y funkciškai priklauso nuo ryšio atributo X tada ir tik tada, kai kiekviena X reikšmė R ryšyje yra susijusi su vienintele R ryšio Y reikšme.**

Pasinaudojant ankstesniu pavyzdžiu, kliento sąskaitos Nr. apibrėžia kliento sąskaitos balansą tik tuo atveju, kai kiekvieną ryšio kliento sąskaitos Nr. atitinka vienas ir vienintelis galimas kliento sąskaitos balansas.

Išraiškoje funkcinė priklausomybė žymima: studento Nr. → studento pavardė. Studento Nr. apibrėžia studento pavardę, nes bet kokiam turimam studento Nr. gali būti viena ir vienintelė studento pavardė. Kairėje rodyklės pusėje esantis atributas (studento Nr.) vadinamas **determinantu**.

Pavyzdžiui, panagrinėkime paprastą lentelę:

1-8 lentelė. Paprastos funkcinės priklausomybės pavyzdys

1 stulpelis	2 stulpelis
A	1
A	1
B	1

Pirmiausia pažiūrėkime, ar 1 stulpelis → 2 stulpelis (ar 1 stulpelis apibrėžia 2 stulpelį). Ar kiekvieną 1 stulpelio reikšmę visuomet atitinka ta pati 2 stulpelio reikšmė? Matome, kad kaskart, kai 1 stulpelyje pasirodo „A“, tos pačios eilutės 2 stulpelyje visuomet būna „1“, o kiekvieną kartą, kai 1stulpelyje yra „B“, tos eilutės 2 stulpelyje visuomet įrašytas „1“. Tai reiškia, kad iš tiesų šiame ribotame eilučių rinkinyje 1 stulpelis apibrėžia 2 stulpelį.

Antra, ar 2 stulpelis → 1 stulpelis? Čia matome, kad kiekvienai 2 stulpelio reikšmei „1“ 1 stulpelyje gali būti arba „A“, arba „B“. Reiškia, 2 stulpelis neapibrėžia 1 stulpelio.

Kartais funkcinės priklausomybės apima atributų grupes. Pavyzdžiui, panagrinėkime lentelę su studentų numeriais, kursų pavadinimais ir įvertinimais. Jei pateiksite tik studento numerį, negalite būti tikri, kad lentelėje rasite vienintelį įvertinimą, kadangi dažnai studentai lanko keletą kursų. Panašiai, jei įrašysite tik kurso pavadinimą, vėl su nurodytu kursu bus susieta keletas įvertinimų, nes tą patį kursą lanko keletas studentų. Šiuo atveju, kad surastumėte konkretų įvertinimą, jums reikės ir studento Nr., ir kurso pavadinimo. Išraiškoje tai rodoma:

(Studento Nr., kurso pavadinimas) → įvertinimas

Taip pat gali būti atvejis, kai vienas atributas gali apibrėžti keletą kitų atributų. Pavyzdžiui, toliau rodoma, kad jūsų vairuotojo teisių numeris apibrėžia ir jūsų pavardę, ir adresą. Lentelėje su šia informacija nurodžius jūsų vairuotojo teisių numerį bus surasta viena ir vienintelė pavardė ir adresas.

Vairuotojo Nr. → pavardė, adresas
(Tas pats kaip vairuotojo Nr. → pavardė, vairuotojo Nr. → adresas)

Yra dvi priklausomybę rodančios išraiškos. Kad palygintume šias dvi išraiškas, panagrinėkime 1-9 lentelėje pateiktą pavyzdį. Šiame pavyzdyje trims stulpeliams suteikti papildomi (*alias*) pavadinimai: A, B ir C, kadangi lengviau perskaityti išraišką A → B. Aptardami šią temą trumpam pasinaudosime tokia A, B, C išraiška.

1-9 lentelė. Išraiškų palyginimo pavyzdys

Studentas (A)	Kursas (B)	Dėstytojas (C)
Tomas	GII-05	Deivas
Bilas	GII-03	Maiklas
Tomas	GII-01	Bredas
Haris	GII-05	Deivas

Šioje lentelėje esančios priklausomybės yra: C → B, ir AB → C. Kaip savarankišką užduotį išnagrinėkite lentelę, kad patikrintumėte, kodėl šiuo atveju yra būtent taip.

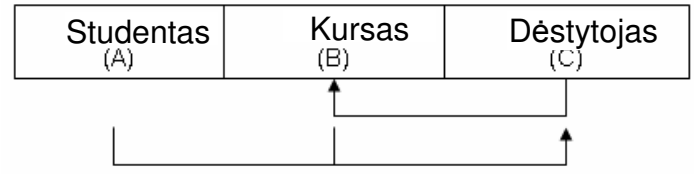
Šios dvi priklausomybės yra:

1) Naudojant „rodyklės“ išraišką, kurią matėme ankstesniuose pavyzdžiuose. 1-9 lentelės priklausomybės atrodys taip:

C → B
AB → C

Išraiška

2) Naudojant priklausomybių diagramas. Tos pačios priklausomybės gali būti vaizduojamos nubrėžus rodykles tarp lentelės stulpelių. Šis metodas leidžia lengviau ir iškart pamatyti, kas vyksta lentelėje, tačiau jį užrašyti trunka ilgiau.



Diagrama

Atlikdami užduotis ir laikydami egzaminus galite naudoti bet kurį metodą, tačiau norėdami nuoseklumo mes, pateikdami paskaitos pavyzdžius, laikysimės išraiškų metodo.

1.2.6 Raktai

Raktas yra vieno ar daugiau atributų grupė, leidžianti be pasikartojimų identifikuoti ryšio eilutę, Raktas ne tik apibrėžia kitą stulpelį, jis apibrėžia visą eilutę (visus stulpelius). Panagrinėkime ryšį, kuriame saugoma studentų aibė, kokiose veiklose jie dalyvauja, ir kiek kainuoja užsiimti tomis veiklomis.

1-10 lentelėje (iš Kroenke, 1998) parodyta situacija, kai studentas gali užsiimti tik viena sporto šaka. Šiuo atveju studento numeri lentelėje yra unikalus, jis gali būti naudojamas kaip raktas. Lentelėje nurodę studento numerį, galėsime be pasikartojimų nustatyti konkrečią lentelės 1-10 lentelė. Studento ir veiklos ryšys. Studento Nr. yra raktaseilutę. Šio ryšio raktas yra tiesiog (studento Nr.).

1-10 lentelė. Studento ir veiklos ryšys. Studento Nr. yra raktas

Studento Nr.	Veikla	Mokestis
100	Slidinėjimas	200 JAV dol.
150	Plaukimas	50 JAV dol.
175	Skvošas	50 JAV dol.
200	Skvošas	50 JAV dol.

1-11 lentelė yra ta pati lentelė, tačiau šįkart studentams leidžiama užsiimti keliomis veiklomis. Šiuo atveju studento numeris jau nėra unikalus. Nurodę studento numerį 100, jau nebegalėsime nustatyti vienintelės to ryšio eilutės. Kad be pasikartojimų nustatytume eilutę, dabar turime įtraukti ir studento numerį, ir veiklą. Šio ryšio raktas yra (studento numeris, veikla). Tai vadinamasis **sudėtinis raktas**, nes jį sudaro daugiau negu vienas atributas.

1-11 lentelė. Studento ir veiklos ryšys. Raktas yra studento numeris ir veikla

Studento Nr.	Veikla	Mokestis
100	Slidinėjimas	200 JAV dol.
150	Plaukimas	50 JAV dol.
175	Skvošas	50 JAV dol.
100	Skvošas	50 JAV dol.

Raktas yra vieno ar daugiau atributų (stulpelių) rinkinys, kuris:

a) be pasikartojimų identifikuoja visus kitus ryšio atributus;

b) yra minimalus atributų rinkinys – derinys, apimantis mažiausią atributų skaičių, kuriuo vis dar galima apibrėžti visą ryšį. Taigi, jei A yra raktas, AB nebus raktas, nes jis nėra minimalus, net jei ir be pasikartojimų nurodytų eilutę.

Kartais gali būti daugiau negu vienas galimas raktas, ir visi jie gali patenkinamai tarnauti kaip ryšio raktai. Jie vadinami **raktais kandidatais**.

1 pavyzdys:

Panagrinėkite studentų ir veiklos 1-10 lentelę ir pabandykite atsakyti į šiuos klausimus. Atsakymai aptariami tolesniame skyriuje.

a) Ar tai ryšys?
b) Ar studento numeris yra raktas?
c) Ar studento numeris yra determinantas?
d) Ar veikla yra raktas?
e) Ar ji determinantas?
f) Ar mokestis yra raktas?
g) Ar jis determinantas?

Sprendimai:

a) Ar tai ryšys?

   Kad tai nustatytume, atsižvelkime į penkis 1.2.4 skyrelio testus. Šiuo atveju mūsų lentelė nepažeidžia nė vieno testo, taigi, taip – ji yra ryšys.

b) Ar studento numeris yra raktas?

   Ar studento numeris be pasikartojimų apibrėžia eilutę? Šiuo atveju studento numeris lentelėje yra unikalus, taigi, taip – jis gali be pasikartojimų apibrėžti eilutę, ir todėl gali būti laikomas raktu.

c) Ar studento numeris yra determinantas?

   Kadangi žinome, kad studento numeris yra raktas, jis taip pat turi būti determinantas, nes gali be pasikartojimų apibrėžti eilutę. Kad surastume eilutę, jis turi apibrėžti visus kitus stulpelius, taigi, turi būti tiesa, kad studento Nr. → veikla ir studento Nr. → mokestis. Taip, studento numeris yra determinantas. Atkreipkite dėmesį, kad **raktai visuomet yra determinantai**.

d) Ar veikla yra raktas?

   Ar tiesa, kad pagal nurodytą veiklą, pavyzdžiui, skvošą, galima be pasikartojimų nustatyti eilutę? Šiuo atveju skvošas randamas du kartus, taigi, negalime unikaliai apibrėžti eilutės. Veikla nėra raktas.

e) Ar ji yra determinantas?

   Ar tiesa, kad pagal nurodytą Veiklą visuomet rasime tą patį studento numerį arba mokestį? Šiuo atveju kiekvienos veiklos kaina visuomet yra tokia pati, taigi, veikla → mokestis. Tačiau veikla neapibrėžia studento Nr.

Tai atributo (veikla), kuris yra determinantas, bet nėra raktas, pavyzdys. Atkreipkite dėmesį, kad nors raktas visuomet yra determinantas, **determinantas ne visuomet yra raktas**.

f) Ar mokestis yra raktas?

Ar tiesa, kad pagal nurodytą mokestį, pavyzdžiui, 50 JAV dol., galima be pasikartojimų nustatyti eilutę? Šiuo atveju ne, nes 50 JAV dol. lentelėje randamas triskart. Mokestis nėra raktas.

g) Ar jis yra determinantas?

Ar tiesa, kad pagal nurodytą mokestį, pavyzdžiui, 50 JAV dol., galima rasti tik vieną studento Nr. arba vieną veiklą? Šiuo atveju 50 JAV dol. randamas su trimis skirtingais studentais ir dviem veiklomis. Taigi, mokestis neapibrėžia nei veiklos, nei studento Nr. Mokestis nėra determinantas.

Raktų nustatymas pagal funkcines priklausomybes

Nagrinėdami funkcines priklausomybes galime nustatyti ryšio raktą. Priklausomybės aprašo ryšio atributų ryšį, taigi, iš šios informacijos galime nustatyti, kaip be pasikartojimų identifikuoti ryšio eilutę.

Šios taisyklės gali padėti supaprastinti arba tvarkyti funkcines priklausomybes:

i. $A \rightarrow A$ (savęs apibrėžimas)

ii. Jei $A \rightarrow BC$, tai $A \rightarrow B$ ir $A \rightarrow C$ (skaidymas)

iii. Jei $A \rightarrow B$ ir $B \rightarrow C$, tai $A \rightarrow C$ (tranzityvumas)

iv. Jei $A \rightarrow BC$ ir $C \rightarrow D$, tai $A \rightarrow BCD$ (sankaupa)

v. Jei $A \rightarrow B$, tai $AC \rightarrow BC$ (išplėtimas)

vi. Jei $A \rightarrow B$ ir $A \rightarrow C$, tai $A \rightarrow BC$ (sąjunga)

vii. Jei A yra raktas, tai AB nėra raktas (minimalus atributų rinkinys)

Naudodamiesi visomis šiomis taisyklėmis bandysime nustatyti minimalų atributų poaibį, kuris gali apibrėžti visą ryšį. Turbūt efektyviausias būdas pademonstruoti, kaip taikomos šios taisyklės, yra išnagrinėti keletą pavyzdžių. Tolesniuose pavyzdžiuose plačiai naudosime tokias išraiškas, kaip R (A, B, C). Tai paprasčiausiai aprašo trijų stulpelių ryšį, o kad būtų paprasčiau, stulpeliai žymimi A, B ir C.

Bendras metodas, kurį taikysime, yra:

1. Patikrinti kiekvieną atskiro atributo raktą (pvz., A, B ir t. t.), kad nustatytume, ar tai raktas.
2. Jei surastas atskiro atributo raktas, nebereikia tikrinti sudėtinių raktų, kadangi jie nebus minimalūs.

3. Jei atskiro atributo raktas nerastas, turime išbandyti visus galimus sudėtinius raktus, kurie apima du atributus (pvz., AB, AC, BC ir t. t.).
4. Jei surastas dviejų atributų raktas, nebereikia tikrinti sudėtinių trijų atributų raktų.
5. Jei dviejų atributų sudėtinis raktas nerastas, patikrinkite trijų atributų sudėtinius raktus (pvz., ABC, ABD, ACD ir t. t.).
6. Ir taip toliau, kol bus surastas raktas kandidatas.

Praktikoje jūsų nebus prašoma tikrinti sudėtinių raktų, kurie apima daugiau negu du atributus, nes sudėtinga patikrinti visas galimas kombinacijas. Šiuo atžvilgiu pasistengsime parinkti tokius pavydžius, kad jie būtų tvarkomi pakankamai lengvai.

2 pavyzdys:

Turime ryšį R (A, B, C, D) su priklausomybėmis A → BC ir B → D. Nustatykite raktą kandidatą (ar kandidatus).

Sprendimas:

Pirmiausia turime patikrinti visus galimus atskirų atributų raktus. Ryšyje R (A, B, C, D) tam reikės patikrinti kiekvieną iš A, B, C ir D, kad matytume, ar jie yra raktai kandidatai.

Bandome A:

A → A	(savęs apibrėžimas)
A → BC, taigi, A → B, A → C	(skaidymas)
A → B, B → D, taigi, A → D	(tranzityvumas)
A → A, A → B, A → C, A → D, taigi, A → ABCD	(sąjunga)

Kadangi A → ABCD, A yra raktas kandidatas.

Tikriname B:

B → B	(savęs apibrėžimas)
B → D	(nurodyta)
B → B, B → D, taigi, B → BD	(sąjunga)

Nėra kitų priklausomybių, kuriose B arba D būtų kairiojoje pusėje, taigi, negalime nustatyti nė vieno kito stulpelio. Kadangi B apibrėžia tik BD, jis neapibrėžia viso ryšio, todėl nėra raktas kandidatas.

Tikriname C:

C neapibrėžia nieko, tik save patį (savęs apibrėžimas), taigi, negalime nustatyti A, B arba D turėdami vien tik C. Todėl C nėra raktas kandidatas.

Tikriname D:

D neapibrėžia nieko, tik save patį (savęs apibrėžimas), taigi, negalime nustatyti A, B arba C turėdami vien tik D. Todėl D nėra raktas kandidatas.

Kadangi atskiro atributo raktas yra minimalus, mums nereikės tikrinti sudėtinių raktų (raktų, apimančių daugiau kaip vieną stulpelį). Todėl vienintelis R (A, B, C, D) raktas kandidatas yra A.

3 pavyzdys:

Turime ryšį R (A, B, C) su priklausomybėmis B → C ir C → B. Nustatykite raktus kandidatus.

Sprendimas:

Pirmiausia turime patikrinti visus galimus atskirų atributų raktus. Ryšyje R (A, B, C) tam reikės patikrinti kiekvieną iš A, B, ir C, kad matytume, ar jie yra raktai kandidatai.

Bandome A:

A neapibrėžia nieko, tik save patį (savęs apibrėžimas), taigi, negalime nustatyti B arba C turėdami vien tik A. Todėl A nėra raktas kandidatas.

Tikriname B:

B → B	(savęs apibrėžimas)
B → C	(nurodyta)
B → B, B → C, taigi, B → BC	(sąjunga)

Turėdami B galime nustatyti BC. Iš B ir C negalime nustatyti A. Nėra priklausomybės, turinčios A dešiniojoje pusėje, taigi, neįmanoma nustatyti A iš B ir (ar) C. Kadangi B apibrėžia tik BC, jis neapibrėžia viso ryšio, todėl nėra raktas kandidatas.

Tikriname C:

C → C	(savęs apibrėžimas)
C → B	(nurodyta)
C → B, C → C, taigi, C → BC	(sąjunga)

Turėdami C galime nustatyti BC. Iš B ir C negalime nustatyti A. Nėra priklausomybės, turinčios A dešiniojoje pusėje, taigi, neįmanoma nustatyti A iš B ir (ar) C. Kadangi C apibrėžia tik BC, jis neapibrėžia viso ryšio, todėl nėra raktas kandidatas.

Taigi, patikrinome visus galimus atskiro atributo raktus ir neradome rakto kandidato. Dabar turime patikrinti visus galimus dviejų atributų raktus. Ryšyje R (A, B, C) tai reiškia, kad turėsime patikrinti AB, AC ir BC.

Tikriname AB:

AB → AB	(savęs apibrėžimas)
AB → AB, taigi, AB → A ir AB → B	(skaidymas)
AB → B, B → C, taigi, AB → C	(tranzityvumas)
AB → A, AB → B, AB → C, taigi, AB → ABC	(sąjunga)

AB → ABC, taigi, AB yra raktas kandidatas.

Tikriname AC:

AC → AC	(savęs apibrėžimas)
AC → AC, taigi, AC → A ir AC → C	(skaidymas)
AC → C, C → B, taigi, AC → B	(tranzityvumas)
AC → A, AC → B, AC → C, taigi, AC → ABC	(sąjunga)

AC → ABC, taigi, AC yra raktas kandidatas.

Tikriname BC:

Turint B ir C nėra jokios galimybės nustatyti A (A niekada nebūna dešiniojoje priklausomybės pusėje), taigi, BC negali būti raktas kandidatas.

Čia radome du kandidatus dviejų atributų sudėtinius raktus (AB ir AC). Nėra reikalo ieškoti trijų atributų sudėtinių raktų, kadangi mūsų turimi dviejų atributų raktai yra minimalūs. Todėl du R (A, B, C) raktai kandidatai yra AB ir AC.

Realybėje, kai labiau įprasite skaityti funkcines priklausomybes ir nustatyti raktus, imsite matyti struktūrą ir suprasite, kad daugeliu atvejų galite atmesti potencialius raktus jų net netikrindami. Pateiktuose pavyzdžiuose išbandėme visus galimus raktus, kad studentai pamatytų, kaip vertinami visi įmanomi raktai. Tačiau yra metodų, kuriuos taikydami galime sumažinti galimų raktų, kuriuos reikia patikrinti, skaičių.

Kad iliustruotume šį principą pasinaudosime ankstesniu 2 pavyzdžiu. 2 pavyzdyje turėjome keturių atributų ryšį, taigi, jei išsamiai tikrintume visas iki dviejų atributų raktų apimančias raktų kombinacijas, iš viso būtų 10 galimų raktų, kuriuos turėtume patikrinti:

Vieno atributo variantai:	A, B, C ir D
Dviejų atributų variantai:	AB, AC, AD, BC, BD ir CD

Yra du principai, kurie turėtų padėti paspartinti raktų tikrinimo procesą.

1. Jei atributo nėra nė vienos priklausomybės kairėje pusėje (t. y. jis nėra determinantas), jis negali būti raktas nei pats vienas, nei derinyje su kitais nedeterminantiniais atributais, nes jis nieko neapibrėžia. Ankstesniame 2 pavyzdyje, ryšyje R (A, B, C, D) turėjome tokias priklausomybes:

   A → BC
   B → D

   Čia C ir D nėra determinantai, taigi, nereikia tikrinti raktų, turinčių vien tik šiuos atributus. Dėl to mums nereikės tikrinti C, D arba CD.

2. Jei atributo nėra nė vienos priklausomybės dešiniojoje pusėje (t. y. jo niekas neapibrėžia), jis privalo būti rakto kandidato (ar raktų kandidatų) dalis (arba pats kaip raktas, arba kaip sudėtinio rakto dalis), nes vienintelis būdas jį apibrėžti – tai įtraukti jį į raktą. Ir vėl. pasinaudoję ankstesniu 2 pavyzdžiu turime priklausomybes:

   A → BC

B → D

A negali apibrėžti niekas kitas (kadangi jo nėra nė vienos priklausomybės dešiniojoje pusėje), taigi, turėtume tikėtis, kad A arba pats turės būti raktas kandidatas, arba turės būti sudėtinio rakto dalis. Pasinaudoję šiuo pastebėjimu, galime atmesti visas A neturinčias kombinacijas. Mums iš tiesų reikės patikrinti tik A, AB, AC ir AD.

Turėkite omenyje, kad jei abejojate, ar nagrinėti raktų kombinaciją, ar ne, visuomet galite ją patikrinti pagal pirmiau apibrėžtas priklausomybių taisykles (tokias, kaip tranzityvumas, sąjunga ir kt.), kurios leidžia nustatyti, ar tai raktas kandidatas.

Profesinėje aplinkoje (t. y., jei jums teks daryti tai „realiame pasaulyje“), svarbu atsižvelgti į tai, jog funkcinių priklausomybių negalima nustatyti vien tik nagrinėjant duomenų rinkinių pavyzdžius. Priklausomybės nustatomos iš semantikos, teorinių modelių arba duomenų bazę kuriančios organizacijos veiklos taisyklių. Nagrinėjant pirmą pavyzdį (studento Nr. kaip raktas – 1-10 lentelė. Studento ir veiklos ryšys. Studento Nr. yra raktas), galima daryti prielaidą, kad studentas gali užsiimti tik viena veikla, tačiau galutinai tai apibrėš organizacijos veiklos taisyklės (t. y. paklauskite ko nors, kas žino!). Gali nebūti jokių kliūčių studentams užsiimti daugiau negu viena veikla, tačiau tai gali neatsispindėti jūsų nagrinėjamame pavyzdyje.

Praktinės užduotys:

Pabandykite patys nustatyti šių ryšių raktus. Atsakymai pateikiami iškart po šių pavyzdžių esančiame skyriuje.

1. Turime ryšį R (A, B, C) su funkcine priklausomybe A → BC, nustatykite raktą kandidatą (ar kandidatus).

2. Turime ryšį R (A, B, C, D) su priklausomybėmis A → B ir C → D, nustatykite raktą kandidatą (ar kandidatus).

Sprendimai:

1. **Turime ryšį R (A, B, C) su funkcine priklausomybe A → BC, nustatykite raktą kandidatą (ar kandidatus).**

Sprendimas:

Pirmiausia turime patikrinti visus galimus atskirų atributų raktus. Ryšyje R (A, B, C) tam reikės patikrinti kiekvieną iš A, B, ir C, kad matytume, ar jie yra raktai kandidatai.

Bandome A:

A → BC	(nurodyta)
A → A	(savęs apibrėžimas)
A → A, A → BC, taigi, A → ABC	(sąjunga)

Kadangi A → ABC, A yra raktas kandidatas, nes apibrėžia visą ryšį.

Tikriname B:

B neapibrėžia nieko, tik save patį (savęs apibrėžimas), taigi, negalime nustatyti A arba C turėdami vien tik B. Todėl B nėra raktas kandidatas.

Tikriname C:

C neapibrėžia nieko, tik save patį (savęs apibrėžimas), taigi, negalime nustatyti A arba B turėdami vien tik C. Todėl C nėra raktas kandidatas.

Kadangi atskiro atributo raktas yra minimalus, mums nereikės tikrinti sudėtinių raktų (raktų, apimančių daugiau kaip vieną stulpelį). Todėl vienintelis R (A, B, C) raktas kandidatas yra A.

2. **Turime ryšį R (A, B, C, D) su priklausomybėmis A → B ir C → D, nustatykite raktą kandidatą (ar kandidatus).**

Sprendimas:

Pirmiausia turime patikrinti visus galimus atskirų atributų raktus. Ryšyje R (A, B, C, D) tam reikės patikrinti kiekvieną iš A, B, C ir D, kad matytume, ar jie yra raktai kandidatai.

Bandome A:

A → A	(savęs apibrėžimas)
A → B	(nurodyta)
A → A, A → B, taigi, A → AB	(sąjunga)

A gali apibrėžti AB, bet negali apibrėžti C arba D. Nėra priklausomybių, pagal kurias turint A arba B apibrėžiama C arba D. Todėl A nėra raktas.

Tikriname B:

B neapibrėžia nieko, tik save patį (savęs apibrėžimas), taigi, negali būti raktas kandidatas.

Tikriname C:

C → C	(savęs apibrėžimas)
C → D	(nurodyta)
C → C, C → D, taigi, C → CD	(sąjunga)

C gali apibrėžti CD, bet negali apibrėžti A arba B. Nėra priklausomybių, pagal kurias turint C arba D apibrėžiama A arba B. Todėl C nėra raktas.

Tikriname D:

D neapibrėžia nieko, tik save patį (savęs apibrėžimas), taigi, negali būti raktas kandidatas.

Taigi, patikrinome visus galimus atskiro atributo raktus ir neradome rakto kandidato. Dabar turime patikrinti visus galimus dviejų atributų sudėtinius raktus. Ryšyje R(A, B, C, D) tai reiškia, kad turėsime patikrinti AB, AC, AD, BC, BD ir CD.

Tikriname AB:

A → B, tačiau tai nepadės – mes tebeturime AB, bet negalime nustatyti C arba D. Nėra priklausomybės, pagal kurią iš A arba B galėtume nustatyti C arba D, taigi, AB nėra raktas.

Tikriname AC:

AC → AC	(savęs apibrėžimas)
AC → AC, taigi, AC → A ir AC → C	(skaidymas)
AC → A, A → B, taigi, AC → B	(tranzityvumas)
AC → C, C → D, taigi, AC → D	(tranzityvumas)
AC → A, AC → B, AC → C, AC → D, taigi, AC → ABCD	(sąjunga)

AC → ABCD, taigi, AC yra raktas kandidatas.

Tikriname AD:

AD → AD	(savęs apibrėžimas)
AD → AD, taigi, AD → A ir AD → D	(skaidymas)
AD → A, A → B, taigi, AD → B	(tranzityvumas)
AD → A, AD → B, AD → D, taigi, AD → ABD	(sąjunga)

AD gali apibrėžti tris iš keturių atributų, tačiau negali apibrėžti C. Nė viena priklausomybė neapibrėžia C, taigi, AD negali būti raktas.

Tikriname BC:

BC → BC	(savęs apibrėžimas)
BC → BC, taigi, BC → B ir BC → C	(skaidymas)
BC → C, C → D, taigi, BC → D	(tranzityvumas)
BC → B, BC → C, BC → D, taigi, BC → BCD	(sąjunga)

BC gali apibrėžti tris iš keturių atributų, tačiau negali apibrėžti A. Nė viena priklausomybė neapibrėžia A, taigi, BC negali būti raktas.

Tikriname BD:

Nei B, nei D neapibrėžia nieko daugiau, taigi, negalime nustatyti A arba C. BD negali būti raktas kandidatas.

Tikriname CD:

C → D, tačiau tai nepadės – mes tegalime nustatyti tik C ir D. Nėra priklausomybės, pagal kurią iš C arba D galėtume nustatyti A arba B, taigi, CD negali būti raktas.

Čia radome vieną dviejų atributų sudėtinį raktą kandidatą (AC). Nėra reikalo ieškoti trijų atributų sudėtinių raktų, kadangi mūsų turimas dviejų atributų raktas yra minimalus. Todėl R (A, B, C, D) raktas kandidatas yra AC.

1.2.7 Normalizavimas

Iki šiol aptarėme, kaip svarbu suprojektuoti duomenų modelį, kuris tenkins organizacijos poreikius, ir užtikrinti, kad būtų tinkamai apibrėžti ryšiai tarp objektų klasių (UML klasių schemos). Šis procesas apibrėžia mūsų duomenų bazės ryšius ir sąsajas tarp ryšių. Ką tik aptarėme ryšius ryšio viduje (funkcinę priklausomybę). Norint užtikrinti gerai parengtą duomenų bazės struktūrą labai svarbu suprasti funkcinę priklausomybę.

Gerai struktūrizuotas ryšys turėtų:

1. Minimizuoti perteklių. Norime užtikrinti, kad duomenų elementai be reikalo nesikartotų. Esant pertekliui neefektyviai naudojama saugojimo vieta ir atliekama paieška.
2. Leisti efektyvią prieigą prie informacijos. Norime būti tikri, kad galėsime greitai surasti ir perskaityti arba atnaujinti duomenų bazės elementus.
3. Leisti įterpimą, modifikavimą ir naikinimą nesukeliant neigiamo pašalinio poveikio arba klaidų ar nenuoseklumų. Tokie neigiami šalutiniai poveikiai vadinami **anomalijomis**, ir duomenų bazėje gali pasitaikyti trijų tipų anomalijos: naikinimo anomalijos, įtraukimo anomalijos ir modifikavimo anomalijos.

Kad iliustruotume šias anomalijas, turime grįžti prie pirmiau aptarto studento ir veiklos ryšio. Šis ryšys dar kartą pakartotas šioje lentelėje:

1-12 lentelė. Studento ir veiklos ryšys

Studento Nr.	Veikla	Mokestis
100	Slidinėjimas	200 JAV dol.
150	Plaukimas	50 JAV dol.
175	Skvošas	50 JAV dol.
200	Skvošas	50 JAV dol.

Naikinimo anomalija

Jei studentas numeriu 100 1-12 lentelėje nuspręs, kad nebenori slidinėti, mes galime ištrinti šią pirmąją ryšio eilutę. Tačiau tai atlikę prarasime ne tik tai, kad studentas numeriu 100 yra slidininkas, bet ir faktą, kad slidinėjimas kainuoja 200 JAV dol. Vienu panaikinimu prarasime du informacijos elementus.

Įtraukimo anomalija

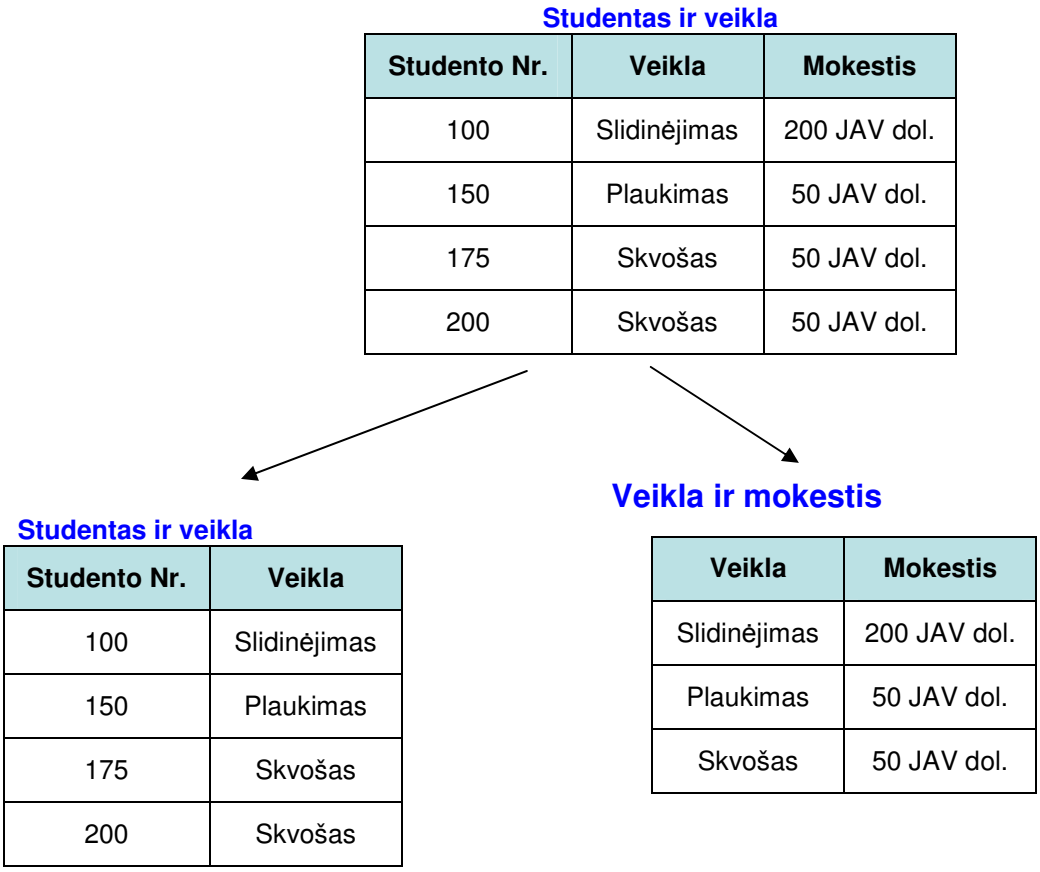
Jei į 1-12 lentelę norėsime įtraukti faktą, kad futbolo treniruotės kainuoja 20 JAV dol., būsime priversti laukti, kol atsiras norintis žaisti futbolą studentas. Negalime įtraukti informacijos apie vieną elementą, kol nepateiksime informacijos apie kitą.

Modifikavimo anomalija

Atkreipkite dėmesį, kad skvošas 1-12 lentelėje įrašytas dukart, nes juo užsiiminėti yra užsiregistravę du studentai. Jei pakeisime faktą, kad skvošo treniruotė kainuoja 50 JAV dol., galbūt, pakeldami kainą iki 60 JAV dol., galime pamiršti pakeisti šią reikšmę abiejose vietose. Tai sukels duomenų konfliktą, kai skvošo kaina dviejose skirtingose vietose bus skirtinga. Tai modifikavimo anomalija.

Visos trys problemos kyla dėl to, kad mūsų studento ir veiklos ryšyje surašytos dvi „temos“ arba dalykai. Studentai, ir kokioje veikloje jie dalyvauja, bei veiklos sritys ir jų kainos. Bandydami sudėti šiuos abu dalykus kartu į vieną ryšį, mes sukeliame neigiamą šalutinį poveikį manipuliuodami duomenimis.

Normalizavimas yra ryšių struktūrizavimo procesas, siekiant minimizuoti šiuos šalutinius poveikius. Normalizavimo esmė yra ta, kad ryšiai turi saugoti tik vieną „temą“. Studento ir veiklos pavyzdyje turėtume suskaidyti studento ir veiklos lentelę į dvi lenteles, kaip parodyta 1-24 pav. 1-24 pav. Studento ir veiklos ryšio normalizavimas



1-24 pav. Studento ir veiklos ryšio normalizavimas

Suskaidžius pradinę lentelę į dvi lenteles išsprendžiamos įterpimo, modifikavimo ir naikinimo anomalijos, taip pat sumažinama saugojimui reikalinga vieta – faktas, kad skvošas kainuoja 50 JAV dol., dabar saugomas tik vieną kartą.

Įterpimas: Galime įterpti naują veiklą kartu su jos mokesčiu net jei nėra ja užsiimti užsiregistravusių studentų.

Modifikavimas: Nėra galimybės pakeisti tik vieną (ne visus) pasikartojantį elementą, nes nėra dubliuotų įrašų.

Naikinimas: Galime laisvai trinti studentus ir jų pasirinktas veiklas nekeisdami veiklų ir jų mokesčių.

Normalizavimas skirstomas į įvairius laipsnius, pagal tai, kaip griežtai ryšys struktūrizuojamas. Pagal struktūros stiprėjimą yra tokios normaliosios formos:

1. Pirmoji normalioji forma (1NF)
2. Antroji normalioji forma (2NF)
3. Trečioji normalioji forma (3NF)
4. *Boyce-Codd* normalioji forma (BCNF)
5. Ketvirtoji normalioji forma (4NF)
6. Penktoji normalioji forma (5NF)
7. Domeno/rakto normalioji forma (DK/NF)

Šios formos yra „įdėtosios“, tai reiškia, kad vieną formą atitinkantis ryšys atitiks ir visas prieš tai esančias formas. Pavyzdžiui, 3 normaliosios formos ryšys automatiškai atitinka 2NF ir 1NF. Kiekviena forma turi vieną ar daugiau testų, pagal kuriuos nustatoma, ar ryšys ją atitinka. Testai taikomi iš eilės, pradedant nuo pirmosios normaliosios formos ir einant prie didėjančio laipsnio arba struktūrinio griežtumo normaliųjų formų. Jei testas netenkinamas (pvz., 3NF testas), ryšys grąžinamas prie paskutinio tenkinamo testo (pvz., 2NF).

Pirmoji normalioji forma

Ryšys yra pirmosios normaliosios formos (1NF), jei jis tenkina ryšio apibrėžimą. Kriterijai, kad lentelė būtų ryšys, yra:

1. Visi tam tikro stulpelio įrašai turi būti to paties tipo.
2. Kiekvienas stulpelis turi nesikartojantį pavadinimą.
3. Stulpelių tvarka nėra svarbi.
4. Lentelėje negali būti identiškų eilučių.
5. Eilučių tvarka nėra svarbi.

Jau esame žiūrėję pavyzdžius, ar lentelė yra ryšys, ar ne, taigi, šis principas turėtų būti jums pažįstamas.

Antroji normalioji forma

Ryšys yra antrosios normaliosios formos (2NF), jei jis tenkina pirmosios normaliosios formos testą, ir:

- turi *vieno atributo raktą*, **arba**
- turi *sudėtinį raktą*, ir *visi nepatenkantys į raktą atributai yra priklausomi nuo viso rakto.*

Šiame apibrėžime vartojama keletas sąvokų, kurias gali tekti paaiškinti:

Vieno atributo raktas: Iš mūsų pokalbio apie raktus tikriausiai atsimenate, kad atskiro atributo raktas yra raktas, apimantis vieną stulpelį (pvz., raktai „studento Nr.“ arba „A“). Jei turime vieno atributo raktą, ryšys automatiškai yra 2NF, ir mums nereikia nagrinėti taisyklės sudėtinio rakto dalies.

Sudėtinis raktas: Ir vėl, iš mūsų pokalbio apie raktus, apibrėžėme sudėtinį raktą kaip apimantį daugiau nei vieną atributą (pvz., raktai (studento Nr., veikla) arba AB). Jei turime sudėtinį raktą, turime žiūrėti į antrąją taisyklės dalį (ir nustatyti, ar visi rakte nesantys atributai priklauso nuo viso rakto).

Nepriklausantys raktui atributai: Nepriklausantys raktui atributai yra tokie, kurie nėra nė vieno rakto dalis. Pavyzdžiui, ryšyje R (A, B, C, D) su raktais kandidatais AB ir AC, vienintelis ne rakto atributas yra D, kadangi A, B ir C visi yra bent vieno rakto kandidato dalys.

Priklausomybė nuo viso rakto: Čia kalbame apie sudėtinius raktus (kadangi atskiro atributo raktas automatiškai atitinka 2NF). Tai, kad nepriklausantys raktui atributai priklauso nuo viso rakto, reiškia, kad negalime turėti nesančio rakte atributo, priklausančio tik nuo rakto dalies (pvz., C yra AC dalis). Pavyzdžiui, jei turime raktą AC ir ne rakto atributą D, jei buvo priklausomybė A → D, tai 2NF testas nebus tenkinamas, kadangi D priklauso nuo A, kuris yra tik rakto (AC) dalis. Kad tenkintų 2NF testą, šiame ryšyje D turėtų priklausyti nuo AC (viso rakto).

Atkreipkite dėmesį, kad jei ten nėra raktui nepriklausančių atributų, o tik sudėtiniai raktai, ryšys tenkina 2NF testą.

Pavyzdžiui, panagrinėkime mūsų pirmiau apibrėžtą studento ir veiklos ryšį su raktu (studento Nr, veikla). Šis ryšys kartojamas 1-13 lentelėje su toliau apibrėžtomis priklausomybėmis ir raktais.

1-13 lentelė. Antrosios normaliosios formos pavyzdys

Studento Nr.	Veikla	Mokestis
100	Slidinėjimas	200 JAV dol.
150	Plaukimas	50 JAV dol.
175	Skvošas	50 JAV dol.
100	Skvošas	50 JAV dol.

Priklausomybės: veikla → mokestis
Raktas: (studento Nr., veikla)

1NF: Ši lentelė yra ryšys, taigi, tenkina 1NF testą.

2NF: Kad taikytume 2NF testą, pirmiausia nuspręsime, ar tai vieno atributo raktas, ar ne. Šiuo atveju turime sudėtinį raktą, nes raktui priklauso du atributai (studento Nr. ir veikla). Kadangi turime sudėtinį raktą, pirmoji taisyklės dalis netaikoma, ir turime tikrinti antrąją dalį.

Kadangi raktas yra (studento Nr., veikla), vienintelis ne rakto atributas lieka Mokestis. Mokestis priklauso nuo Veiklos, kuri yra tik sudėtinio rakto dalis, taigi, šis ryšys netenkina 2NF testo. Jis grąžinamas prie paskutinio išlaikyto testo, pagal kurį nustatyta 1NF, taigi, galima sakyti, kad šis ryšys yra pirmosios normaliosios formos.

Trečioji normalioji forma

Ryšys yra trečiosios normaliosios formos (3NF), jei jis tenkina antrosios normaliosios formos testą, ir

- *neturi tranzityviųjų priklausomybių*

Tranzityviosios priklausomybės: Ankstesniame šio dokumento skyrelyje (1.2.6 Raktai) aptarėme priklausomybių tranzityvumo savybę (pvz., jei A → B ir B → C, tai A → C). Jei turime priklausomybę, kuriai galioja tranzityvi situacija, ryšys netenkina trečiosios normaliosios formos testo.

- Gera praktinė taisyklė ieškoti tranzityviųjų priklausomybių yra tokia, kad **neturi būti priklausomybių, kuriose raktui nepriklausantį atributą apibrėžia kitas raktui nepriklausantis atributas** (t. y. abi funkcinės priklausomybės pusės nėra raktai). Jei taip yra, turime tranzityviąją priklausomybę, ir ryšys netenkina 3NF testo.

Pavyzdžiui, panagrinėkime Studento-Veiklos ryšį tuo atveju, kai studentas gali užsiregistruoti tik vienai veiklai. Ryšys gali būti apibrėžtas kaip:

Ryšys:	VEIKLOS (Studento Nr., Veikla, Mokestis)
Priklausomybės:	Studento Nr. → Veikla Studento Nr. → Mokestis Veikla → Mokestis
Raktai	Studento Nr.

1NF: Šiuo atveju nustatėme, kad tai ryšys, taigi, žinome, kad jis tenkina 1NF testą.

2NF: Jis turi vieno atributo raktą (studento Nr), taigi, galime iškart nustatyti, kad jis tenkina ir 2NF testą.

3NF: Kadangi studento Nr. yra raktas, jis apibrėžia ir veiklą, ir mokestį. Kadangi Veikla → Mokestis, turime tranzityviąją priklausomybę, nes Studento Nr. → Veikla ir Veikla → Mokestis. Tikrindami pagal mūsų praktinę taisyklę matome, kad Veikla → Mokestis yra priklausomybė, kurios abiejose pusėse nėra raktų, todėl šiuo būdu ji irgi netenkinama. Todėl šis ryšys yra antrosios normaliosios formos.

***Boyce-Codd* normalioji forma**

Ryšys yra *Boyce-Codd* normaliosios formos (BCNF), jei jis tenkina trečiosios normaliosios formos testą, ir

- *kiekvienas determinantas yra raktas kandidatas*

Determinantas: determinantas yra kairioji funkcinės priklausomybės pusė.

Raktas kandidatas: raktas kandidatas yra vienas iš keleto galimų raktų.

Kad kiekvienas determinantas būtų raktas kandidatas, visos funkcinių priklausomybių kairiosios pusės turi būti raktai kandidatai. Jei yra determinantas, kuris nėra raktas kandidatas, ryšys netenkina BCNF testo.

Pavyzdžiui, panagrinėkime tokį ryšį:

Ryšys:	R (A, B, C)
Priklausomybės:	B → C
	C → B
Raktai	AB, AC

1NF: Ir vėl, mums nurodyta, kad R (A, B, C) yra ryšys, taigi, žinome, kad jis atitinka bent 1NF.

2NF: Raktai kandidatai yra sudėtiniai, taigi, pirmoji taisyklės dalis netaikoma. Turime nustatyti, ar visi rakte nesantys atributai priklauso nuo viso rakto. Raktai kandidatai yra AB ir AC, taigi, ten nėra raktui nepriklausančių atributų. Kadangi nėra raktui nepriklausančių atributų 2NF testas tenkinamas.

3NF: Čia nėra tranzityviųjų priklausomybių, ir patikrinimui – čia nėra priklausomybių, kuriose nepriklausantys raktui atributai būtų abiejose priklausomybės pusėse (tiek B, tiek C patenka į raktus kandidatus AB ir AC). Todėl šis ryšys tenkina 3NF.

BCNF: Tiek B, tiek C abu yra determinantai, tačiau nė vienas nėra raktas kandidatas, taigi, šis ryšys neišlaiko BCNF testo. Todėl šis ryšys yra trečiosios normaliosios formos.

Aukštesnės normaliosios formos

Už *Boyce-Codd* normaliosios formos yra 3 aukštesnės normaliosios formos: ketvirtoji normalioji forma, penktoji normalioji forma ir domeno-rakto normalioji forma. Kiekviena iš jų yra vis abstraktesnė, o BCNF panaikina praktiškai visas anomalijas. Dėl to, ir kadangi šis kursas skirtas pateikti duomenų struktūrų pagrindus, o ne išsamų normalizavimo supratimą. mes nenagrinėsime šių aukštesniųjų formų.

Praktinės užduotys:

Savarankiškai pabandykite išspręsti šiuos normaliųjų formų pavyzdžius. Atsakymai pateikiami iškart po šių pavyzdžių esančiame skyriuje.

1. Šioje lentelėje nustatykite:
    a) visas funkcines priklausomybes (įskaitant dviejų atributų priklausomybes)
    b) visus raktus kandidatus (įskaitant dviejų atributų raktus)

c) jos normaliąją formą

A	B	(C)
A	1	x
B	2	y
C	1	z
A	1	z
A	1	y
B	2	z

2. Ryšys: R (A, B, C)
   Priklausomybės: C → B
   AB → C

   Nustatykite:
   a) raktus kandidatus (įskaitant dviejų atributų raktus)
   b) jos normaliąją formą

3. Ryšys: R (A, B, C, D, E, F)
   Priklausomybės: A → B
   C → D
   AC → E
   AC → F

   Nustatykite:
   a) raktus kandidatus (įskaitant dviejų atributų raktus)
   b) jos normaliąją formą

4. Šioje lentelėje nustatykite:
   a) visas funkcines priklausomybes (iki, ir įskaitant dviejų atributų priklausomybes)
   b) raktus kandidatus (įskaitant dviejų atributų raktus)
   c) jos normaliąją formą

A	B	C
1	a	1001
2	b	1001
3	a	2001
4	a	2001
5	a	3001

5. Ryšys: R (A, B, C, D)
   Priklausomybės: C → D
   C → A

B → C

Nustatykite:

a) raktus kandidatus (įskaitant dviejų atributų raktus)
b) jos normaliąją formą

Sprendimai:

1. **Šioje lentelėje nustatykite:**
   a) **visas funkcines priklausomybes (iki, ir įskaitant dviejų atributų priklausomybes)**
   b) **raktus kandidatus (įskaitant dviejų atributų raktus)**
   c) **jos normaliąją formą**

A	B	C
A	1	x
B	2	y
C	1	z
A	1	z
A	1	y
B	2	z

FP: A → B

Raktai: **AC** (Nerodysime, kodėl visi kiti galimi raktai nėra raktai, tik kodėl raktas yra šis)

AC → AC (savęs apibrėžimas)
AC → AC, taigi, AC → A ir AC → C (skaidymas)
AC → A, A → B, taigi, AC → B (tranzityvumas)
AC → A, AC → B, AC → C, taigi, AC → ABC (sąjunga)

1NF: Tenkina visus ryšio reikalavimus, taigi, 1NF.

2NF: Nėra vieno atributo rakto.
Sudėtinis raktas, taigi, rakte nesantys atributai turi priklausyti nuo viso rakto.
Nepriklausantis raktui atributas yra B, kadangi AC yra raktas.
B priklauso tik nuo A, kuris yra tik rakto (AC) dalis. Netenkina 2NF

3NF: n/a

BCNF: n/a

Rezultatas: 1NF

2. **Ryšys:** **R (A, B, C)**
   Priklausomybės: **C → B**
   AB → C
   Nustatykite:
   a) **raktus kandidatus (įskaitant dviejų atributų raktus)**
   b) **jos normaliąją formą**

FP: C → B
AB → C (nurodyta)

Raktai: **AB, AC** (Nerodysime, kodėl visi kiti galimi raktai nėra raktai, tik kodėl raktai yra šie)

AC → AC	(savęs apibrėžimas)
AC → AC, taigi, AC → A ir AC → C	(skaidymas)
AC → C, C → B, taigi, AC → B	(tranzityvumas)
AC → A, AC → B, AC → C, taigi, AC → ABC	(sąjunga)

AB → AB	(savęs apibrėžimas)
AB → C	(nurodyta)
AB → AB, AB → C, taigi, AB → ABC	(sąjunga)

1NF: Pateikta kaip ryšys, taigi, tenkina 1NF.

2NF: Nėra vieno atributo rakto.
Sudėtinis raktas, taigi, rakte nesantys atributai turi priklausyti nuo viso rakto.
Nėra rakte nesančių atributų, kadangi AB, AC yra raktai kandidatai, taigi, tenkina 2NF.

3NF: Ar nėra tranzityviųjų priklausomybių? AB → C ir C → B panaši į tranzityviąją priklausomybę, tačiau iš tiesų tokia nėra, kadangi C → B tik dar kartą apibrėžia B, kuris yra AB → C dalis (t. y. C neapibrėžia nieko, kas jau nebūtų AB dalis). Kitas būdas tai patikrinti yra ieškoti priklausomybių, kurių abiejose FP pusėse nėra raktų. Kadangi ten nėra raktams nepriklausančių atributų, tokios situacijos būti negali, taigi tranzityviųjų priklausomybių nėra.

BCNF: Ar visi determinantai yra raktai kandidatai? Ne, kadangi turime determinantą (C), kuris nėra raktas kandidatas. Šis ryšys netenkina BCNF.

Rezultatas: 3NF

3. Ryšys: R(A, B, C, D, E, F)
Priklausomybės: A → B
C → D
AC → E
AC → F

Nustatykite:

a) raktus kandidatus (įskaitant dviejų atributų raktus)
b) jo normaliąją formą

FP: A → B
C → D
AC → E
AC → F

Raktai: **AC**

AC → AC	(savęs apibrėžimas)
AC → AC, taigi AC → A ir AC → C	(skaidymas)
AC → A, A → B, taigi, AC → B	(tranzityvumas)
AC → C, C → D, taigi, AC → D	(tranzityvumas)
AC → E	(nurodyta)
AC → F	(nurodyta)

AC → A, AC → B, AC → C, AC → D,
AC → E, AC → F, taigi, AC → ABCDEF (sąjunga)

1NF: Pateikta kaip ryšys; tenkina 1NF.
2NF: Nėra vieno atributo rakto.
Ne rakto atributai B, D, E ir F (raktas yra AC)
Tiek B, tiek D priklauso tik nuo rakto dalių, taigi, šis ryšys netenkina 2NF.
3NF: n/a
BCNF: n/a
Rezultatas: 1NF

4. Šioje lentelėje nustatykite:
a) visas funkcines priklausomybes (iki, ir įskaitant dviejų atributų priklausomybes)
b) raktus kandidatus (įskaitant dviejų atributų raktus)
c) jo normaliąją formą

A	B	C
1	a	1001
2	b	1001
3	a	2001
4	a	2001
5	a	3001

FP: A → B
A → C
Raktai: **A**
A → A (savęs apibrėžimas)
A → B (nurodyta)
A → C (nurodyta)
A → A, A → B, A → C, taigi, A → ABC (sąjunga)

1NF: Tenkina visus ryšio reikalavimus, taigi, 1NF.
2NF: Vieno atributo raktas; tenkina 2NF.
3NF: Nėra tranzityviųjų priklausomybių. Atliekant dvigubą patikrinimą nerandama funkcinių priklausomybių, kurių abi pusės nėra raktai. Tenkina 3NF.
BCNF: Ar visi determinantai yra raktai kandidatai? Čia vienintelis determinantas yra A, kuris taip pat yra raktas, taigi, BCNF tenkinama.
Rezultatas: BCNF

5. Ryšys: R (A, B, C, D)
Priklausomybės: C → D
C → A
B → C

Nustatykite:
a) raktus kandidatus (įskaitant dviejų atributų raktus)
b) jo normaliąją formą

FP:	C → D	
	C → A	
	B → C	
Raktai:	**B**	
	B → B	(savęs apibrėžimas)
	B → C	(nurodyta)
	B → C, C → A, taigi, B → A	(tranzityvumas)
	B → C, C → D, taigi, B → D	(tranzityvumas)
	B → A, B → B, B → C, B → D, taigi, B → ABCD	(sąjunga)
1NF:	Nurodyta; tenkina 1NF.	
2NF:	Vieno atributo raktas; tenkina 2NF.	
3NF:	Yra tranzityviųjų priklausomybių.	
	Iš tiesų dvi iš jų: B → C, C → A; B → C, C → D	
	Netenkina 3NF.	
BCNF:	n/a	
Rezultatas:	2NF	

1.2.8 Denormalizavimas

Normalizavimu stengiamasi sumažinti perteklių ir išvengti anomalijų. Todėl jis turėtų sudaryti dalį kiekvienos duomenų bazės projektavimo proceso. Iš esmės, atliekant normalizavimo procesą šių tikslų siekiama skaidant lenteles į dvi ar daugiau lentelių, kurių kiekviena turi savą „temą“ arba dalyką. Ryšyje turime vieną temą, jei nėra kitų priklausomybių, išskyrus apimančias raktus (tai *Boyce-Codd* normaliosios formos apibrėžimas: visi determinantai yra raktai kandidatai).

Tačiau kaskart, kai ryšys suskaidomas į daugiau negu vieną duomenų bazės lentelę, norint sujungti informaciją į vieną vietą analizei arba pateikimui, reikės papildomos įvesties į diską ir išvesties iš jo bei apdorojimo. Šios papildomos sąnaudos priimtinos, kai svarbus duomenų vientisumas, tačiau yra atvejų, kai papildomos normalizavimo sąnaudos (efektyvumo požiūriu) nėra pagrįstos laimimu duomenų vientisumu. Tokiais atvejais ryšiai sąmoningai **denormalizuojami**, arba paliekami vieno ryšio forma, net jei dėl to lieka žemesnės normaliosios formos. Denormalizavimas yra projektinis kompromisas tarp efektyvumo ir vientisumo, ir jo nauda priklausys nuo to, kaip duomenų bazė bus naudojama. Paprastai denormalizavimas dėl potencialių denormalizuotos duomenų bazės problemų (anomalijų) turėtų būti naudojamas atsargiai.

Nors yra keletas pakankamai aiškių atvejų, kai denormalizavimas turi prasmės. Pavyzdžiui, panagrinėkime labai paprastą lentelę su tokiais laukais:

KLIENTAI

Kliento ID	Kliento pavardė	Miestas	Šalis	Pašto kodas

Savaime suprantama, kad tarp kliento ir jo gyvenamosios vietos yra ryšys. Tai aišku. Tačiau vidinė priklausomybė pašto kodas → (miestas, šalis) reiškia, kad šioje lentelėje turime ne vieną dalyką. Taigi, turime trijų mums žinomų anomalijų galimybę:

Įterpimas: negalime įtraukti pašto kodo ir miesto bei šalies, kurioje jis yra, jei toje vietoje nėra klientų.
Naikinimas: jei ištrinsime klientą, prarasime miestą ir šalį, kuriuose yra pašto kodas.
Modifikavimas: jei pakeisime pašto kodo miestą arba šalį tik vienoje vietoje, bet ne kitoje, atsiras reikšmių konflikto galimybė.

Jei stengiamės bet kokia kaina pasiekti projektinį grynumą, suskaidysime mūsų pradinį ryšį į du ryšius:

KLIENTAI

Kliento ID	Kliento pavardė	Pašto kodas

PAŠTO KODAI

Pašto kodas	Miestas	Šalis

Tokiu būdu galime tvarkyti visus pašto kodus ir jų vietas (miestus ir šalis) viename ryšyje, o klientus ir jų buvimo vietas – kitame. Tai teoriniu požiūriu teisinga realizacija.

Tačiau daugeliu atvejų masyvus visų pašto kodų ir miestų bei šalių, kuriose jie yra, sąrašas organizacijai neturi jokios realios vertės. Viskas, ko joms iš tiesų reikia, yra klientų ir klientų buvimo vietų lentelė. Padidėję šio pašto kodų sąrašo priežiūros sąnaudos bei didesnės apimties apdorojimas ir įvestis/išvestis, kurių duomenų bazei reikia tvarkyti suskaidytas lenteles, nesukuria jokios realios grįžtamosios vertės. Įterpimo ir naikinimo anomalijos yra nereikšmingos. Pavyzdžiui, jei prarasime pašto kodą ir miestą su šalimi, kur jis yra, tai neturės jokios įtakos mūsų veiklai. Tikriausiai sutiksime gyventi su galimomis modifikavimo anomalijomis, kaip efektyvumo, pasiekto sudėjus visus šiuos atributus į vieną lentelę, šalutiniu poveikiu.

Šiuo atveju yra prasmės palikti tai struktūrizuota kaip vienas ryšys. Laimėtas vientisumas nevertas papildomų skaidymo į du ryšius sąnaudų. Šiuo atveju tikriausiai priimsime projektinį sprendimą denormalizuoti šį ryšį.

Literatūra

- Blaha, M., and Premerlani, W. *Object-Oriented Modeling and Design for Database Applications*. Prentice-Hall, 1998.
- Date, C.J. *An Introduction to Database Systems*. Addison-Wesley, 2000.
- Date, C.J. *The Database Relational Model: A Retrospective Review and Analysis*. Addison-Wesley, 2001.
- Elmasri, R., and Navathe, S.B. *Fundamentals of Database Systems*. Addison-Wesley, 2000.
- Kroenke, D.M. *Database Processing Fundamentals, Design and Implementation*. Prentice-Hall, 1998.
- Naiburg, E.J., and Maksimchuk, R.A. *UML for Database Design*. Addison-Wesley, 2001.
- Rob, P., and Coronel, C. *Database Systems: Design, Implementation and Management*. Thomson Learning, 2000.
- Satzinger, J.W., Jackson, R.B., and Burd, S.D. *Systems Analysis and Design in a Changing World.* Thompson Learning, 2000.
- Shekhar, S., and Chawla, S. *Spatial Databases: A Tour*. Prentice-Hall, 2003.
- Zeiler, M. *Modeling Our World*. Redlands, CA: ESRI Press, 1999.

2 Praktinis įgyvendinimas

Šioje dalyje nagrinėjamas perėjimas nuo teorijos prie praktikos. Ankstesniuose užsiėmimuose sužinojome, kad klasių diagramos yra priemonė saugomų duomenų koncepciniams modeliams kurti, o raktai ir normalizavimas leidžia nustatyti, ar gautas sąryšinis duomenų modelis yra geras. Šioje dalyje aptarsime, kaip šiuos duomenų modelius įgyvendinti praktiškai.

1 temoje nagrinėjamas duomenų bazių lentelių kūrimas pagal klasių diagramas, įskaitant ir raktų naudojimą ryšiams tarp lentelių sudaryti. 2 temoje išsamiai nagrinėjamos *ArcMap* programos ir jos pirmtakių naudojamos duomenų struktūros. 3 temoje išsamiai nagrinėjamas geoduomenų bazės modelis bei įvairūs elementai, padedantys modeliuoti erdvinius ir neerdvinius ryšius. Paskutinėje temoje trumpai aptariamas naujos geoduomenų bazės kūrimo procesas – nuo loginio projekto iki užbaigtos, tinkamos naudoti duomenų bazės.

Dalies turinys

- Klasių diagramų praktinis įgyvendinimas
- ESRI duomenų struktūros
- Geoduomenų bazės elementai
- Praktinis geoduomenų bazės įgyvendinimas

Yra sukurta daug erdviniams duomenims saugoti tinkamų duomenų bazių, tarp jų *Oracle Spatial*, IBM *DB2 Spatial Extender* ir *Informix Spatial DataBlade*. Geoduomenų bazės įgyvendinimo sąvokoms iliustruoti pasirinkome ESRI (*Environmental Systems Research Institute*) sukurtą geoduomenų bazę, skirtą *ArcMap* GIS programai. Ši duomenų bazė pasirinkta todėl, kad praktinėse šio kurso dalyse dirbama su *ArcMap* programa, ir todėl, kad ESRI geoduomenų bazė pagrįstai laikoma viena geriausių savo galimybėmis ir yra viena daugiausiai naudojamų.

2.1 Klasių diagramų praktinis įgyvendinimas

Kol kas šiame kurse nagrinėjome teorinę geoduomenų bazių projektavimo dalį. Kai jau mokame sumodeliuoti duomenų bazę ir nustatyti, ar sukurtas modelis geras, galime imti ir paversti šį modelį tikrais reliacinės duomenų bazės ryšiais (lentelėmis). Šioje temoje nagrinėsime, kaip klasių diagramų elementai, pavyzdžiui, objektų klasės ir asociacijos tampa ryšiais. Be to, išsiaiškinsime, kaip šiuose naujuose ryšiuose raktais sudaryti norimas asociacijas.

2.1.1 Praktinis klasių įgyvendinimas

UML klasių diagramų klases paversti duomenų bazės lentelių apibrėžimais elementaru. Kiekviena diagramos klasė tampa duomenų bazės lentele, o kiekviena klasės savybė – lentelės lauku. Paprastai lentelės pavadinimas būna toks pat, kaip ir klasės.

Pavyzdžiui, įsivaizduokime paveikslėlyje apibrėžtą KLIENTŲ klasę. Atkreipkite dėmesį, kad žymėjimas truputį pakeistas – klasės raktas yra pabrauktas, kad vėliau galėtume sukurti asociacijas.

Norėdami šią klasę įgyvendinti praktiškai, paprasčiausiai sukursime lentelę su klasių savybių laukais. Galutinės duomenų bazės ryšių/lentelių apibrėžimus rašysime taip:

KLIENTAS(KlientoNr, KlientoVardPav, Adresas, Miestas, Šalis, PaštoIndeksas, TelNumeris)

Šis žymėjimas primena duomenų apibrėžimo kalbą (DDL – *Data Definition Language*), kuria galima rašyti *Oracle*, *SQL Server* ir kitų bazių lentelių apibrėžimus; skirtumas tas, kad nenurodomi duomenų tipai ir apribojimai, kurių reikia lentelei užbaigti. *Oracle RDBVS* KLIENTŲ klasės apibrėžimas DDL kalba atrodytų taip:

```
/* Lentelės apibrėžimas */
CREATE TABLE KLIENTAS (
        KLIENTONR NUMBER(8),
        KLIENTOPVARDPAV VARCHAR2(30),
        ADRESAS VARCHAR2(50),
        MIESTAS VARCHAR2(20),
          PASTOINDEKSAS VARCHAR2(6),
        TELNUMERIS NUMBER(10)
```

```
);

/* Pagrindinio rakto apribojimas */
ALTER TABLE KLIENTAS (
CONSTRAINT KLIENTO_PR
PRIMARY KEY (KLIENTOPVARDPAV)
);
```

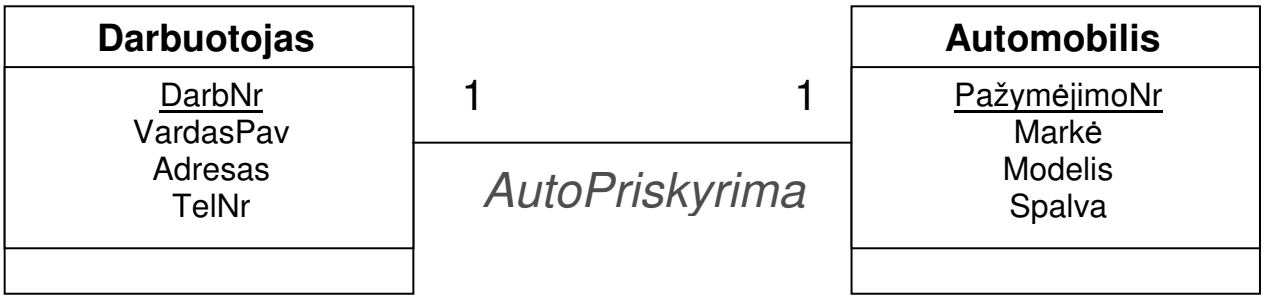
Mes naudosime ne DDL, o supaprastintą žymėjimą dėl dviejų priežasčių:

1. Dabar mūsų tikslas yra suprasti, kaip lentelės kuriamos ir raktais susiejamos viena su kita, o ne gilintis į smulkmenas, susijusias su duomenų tipais ir apribojimais.
2. Fizinės duomenų bazės įgyvendinimo detalės, įskaitant duomenų tipus ir apribojimus, priklauso nuo konkrečios RDBVS programos. Pavyzdžiui, laukas, kurio reikšmės yra tekstinės (pvz., KlientoVardPav), Oracle bazėje bus VARCHAR2 tipo, SQL Server – NVARCHAR tipo, o MS Access – TEXT tipo. Kol kas stengsimės išlaikyti duomenų bazę pakankamai abstrakčią, kad nepriklausytume nuo konkretaus jos varianto.

Nors klases įgyvendinti pakankamai paprasta, paversti klasių diagramos asociacijas lentelių apibrėžimais truputį sudėtingiau. Kiekvieno tipo asociacija (1:1, 1:M ir M:M) įgyvendinama šiek tiek kitaip. Toliau visas jas išnagrinėsime iš eilės.

2.1.2 „Vienas su vienu“ (1:1) asociacijų praktinis įgyvendinimas

Kaip 1:1 asociacijos pavyzdį dar kartą panagrinėsime šio kurso 1 dalyje aptartą darbuotojų-automobilių asociaciją. Ja vaizduojamas ryšys, kai kiekvienas organizacijos darbuotojas gauna atskirą automobilį darbo reikalams. Tai ryšys 1:1, nes darbuotojai automobilių nesidalija.



2-1 pav. Automobilių skyrimo asociacija

Kad sukurtume ryšį, pirmiausia sukursime dvi lenteles – DARBUOTOJŲ ir AUTOMOBILIŲ. Tada vienos lentelės raktą įkelsime į kitą lentelę kaip papildomą lauką, nurodantį, kuris darbuotojas kuriuo automobiliu važinėja. Šis laukas vadinamas **išoriniu raktu**. Toliau pateiktuose pavyzdžiuose išoriniai raktai užrašyti *kursyvu*.

1:1 asociacijas galima įgyvendinti dviem būdais. Galima į automobilių lentelę įrašyti darbuotojų lentelės raktą (DarbNr), arba į darbuotojų lentelę įrašyti automobilių lentelės raktą, kaip pavaizduota žemiau:

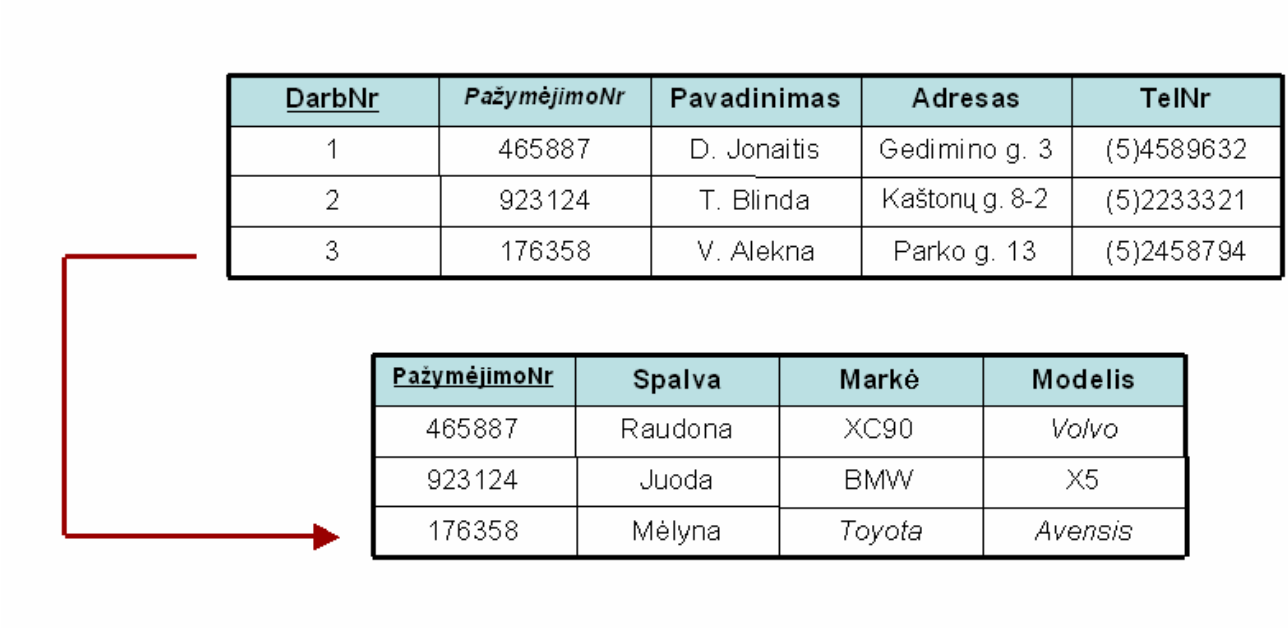
DARBUOTOJAS(DarbNr, VardasPav, Adresas, Telefonas)

AUTOMOBILIS(PažymėjimoNr, *DarbNr*, Markė, Modelis, Spalva)

ARBA

DARBUOTOJAS(DarbNr, *PažymėjimoNr*, VardasPav, Telefonas)
AUTOMOBILIS(PažymėjimoNr, Markė, Modelis, Spalva)

1:1 ryšiui teisingi abu variantai. 1:1 ryšiai – vienintelis ryšių tipas, kurį galima sudaryti dviem teisingais būdais. Įkėlus duomenis į fizines lenteles, antruoju būdu sudarytas ryšis atrodys kaip 2-2 pav. pavaizduotos lentelės.



DarbNr	*PažymėjimoNr*	Pavadinimas	Adresas	TelNr
1	465887	D. Jonaitis	Gedimino g. 3	(5)4589632
2	923124	T. Blinda	Kaštonų g. 8-2	(5)2233321
3	176358	V. Alekna	Parko g. 13	(5)2458794

PažymėjimoNr	Spalva	Markė	Modelis
465887	Raudona	XC90	*Volvo*
923124	Juoda	BMW	X5
176358	Mėlyna	*Toyota*	*Avensis*

2-2 pav. Praktinio automobilių skyrimo asociacijos įgyvendinimo pavyzdys

Šiuo atveju DARBUOTOJŲ lentelėje sukurtas papildomas laukas – išorinis raktas *PažymėjimoNr*. Imdama duomenis iš šių dviejų lentelių, RDBVS programa susietos lentelės reikšmių „ieško“ pagal išorinį raktą. Pavyzdžiui, jei norime nustatyti, koks automobilis skirtas Virgilijui Aleknai, randame V. Aleknos įrašą darbuotojų lentelėje. Iš šios lentelės paimame jo išorinį raktą – *PažymėjimoNr*, kurio reikšmė yra „176358“. Tada šią reikšmę (176358) galime rasti automobilių lentelėje. Pagal ją nustatysime, kad V. Alekna vairuoja mėlyną „Toyotą“. Taip išoriniu raktu galima susieti dvi lenteles.

Jau minėjome, kad abu variantai funkciniu atžvilgiu vienodi, tačiau jei ketinate dažnai atlikti tam tikro tipo paiešką, vienas iš jų gali būti spartesnis. 2-1 lentelėje šie du variantai palyginti ir aprašyti veiksmai, atliekami vykdant konkrečias duomenų paieškos operacijas.

2-1 lentelė. 1:1 ryšio variantų palyginimas

	Pirmas variantas (AUTOMOBILIS turi abu raktus)	Antras variantas (Darbuotojas turi abu raktus)
Automobilis žinomas, rasti darbuotoją	▪ Imti pažymėjimo numerį, jo eilutėje AUTOMOBILIŲ lentelėje surasti DarbuotojoNr ▪ DARBUOTOJŲ lentelėje rasti darbuotoją	▪ Imti pažymėjimo numerį, toje pačioje DARBUOTOJŲ lentelėje perskaityti informaciją apie darbuotoją
Darbuotojas žinomas, rasti automobilį	▪ AUTOMOBILIŲ lentelėje surasti darbuotojo numerį ir perskaityti informaciją apie automobilį	▪ Imti darbuotojo numerį, jo eilutėje DARBUOTOJŲ lentelėje surasti pažymėjimo numerį ▪ Rasti šį pažymėjimo numerį AUTOMOBILIŲ lentelėje ir perskaityti informaciją apie automobilį

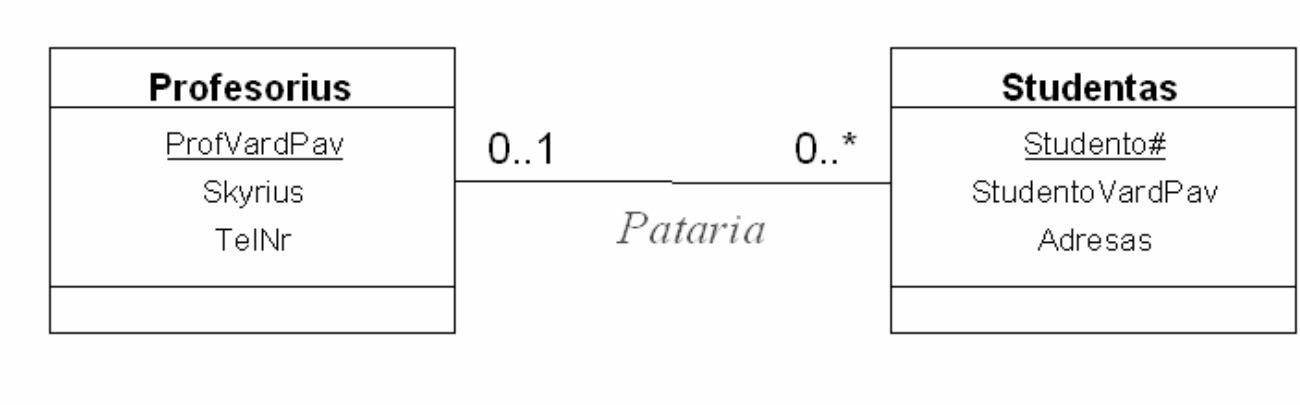
Pavyzdžiui, jei manote, kad dažniausiai teks ieškoti, koks automobilis skirtas konkrečiam darbuotojui, logiška pasirinkti pirmąjį variantą (AUTOMOBILIS turi abu raktus), nes tada reikės perpus mažiau „skaitymo“ iš lentelių operacijų.

2.1.3 „Vienas su daug“ (1:M) asociacijų praktinis įgyvendinimas

1:M asociaciją paversti lentele paprasčiau negu 1:1 vien dėl to, kad šią asociaciją įgyvendinti galima vieninteliu būdu. Šiuo atveju nereikia rinktis variantų ir svarstyti, kokios transakcijos bus dažniausios.

Įgyvendinant 1:M, paprasčiausiai imamas asociacijos „1” pusės raktas ir įrašomas į „Daug“ pusės lentelę kaip išorinis raktas. „1” pusė kartais vadinama pirmine (parent) lentele, o „M“ pusė – antrine (*child*) lentele. Jei prisimenate, ši terminologija buvo vartojama aprašant hierarchinį duomenų bazių modelį. Antrinė lentelė visada turi papildomą lauką – sudarant jos struktūrą, pridedamas išorinis raktas.

Pavyzdžiui, įsivaizduokite profesoriaus-studentų asociaciją. Tai gali būti studentų ir jų darbo vadovų ryšys. Šiuo atveju studentui fakultetas skirs ne daugiau negu vieną vadovą, o profesorius gali vadovauti keliems studentams.



2-3 pav. Profesorių-studentų asociacija

Kaip ir kurdami 1:1 asociacijas, sukursime šias dvi klases vaizduojančias lenteles bei pridėsime klasių savybes atitinkančius laukus. Tada pirminės klasės (Profesorius; „1“ asociacijos pusė) raktą *ProfVardPav* įrašysime į antrinę klasę (Studentas; „M“ asociacijos pusė) kaip išorinio rakto lauką. Gautas ryšis atrodys taip:

PROFESORIUS(ProfVardPav, Skyrius, TelNr)
STUDENTAS(Studento#, *ProfVardPav*, StudentoVardPav, Adresas)

Fizinėse lentelėse rezultatas gali būti panašus į pavaizduotą 2-2 lentelėje, STUDENTAS turės naują išorinio rakto lauką. Iš lentelės matyti, jog 1:M įgyvendinama todėl, kad *ProfVardPav* (pvz., Mykolas) STUDENTŲ lentelėje gali kartotis – tai reiškia, kad Mykolas vadovauja keliems studentams.

2-2 lentelė. Praktiškai įgyvendinta profesorių-studentų asociacija

PROFESORIUS

ProfVardPav	**Skyrius**	**TelNr**
Mykolas	GEOG	9876
Tomas	IT	1234
Juozas	GEOG	5542

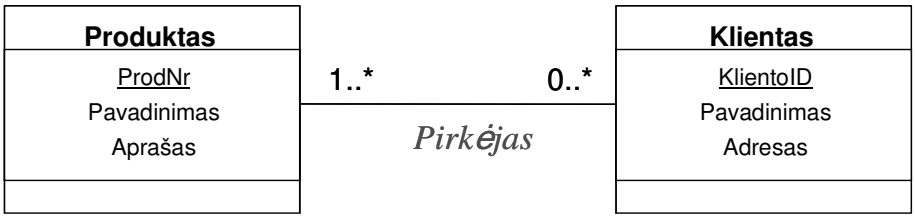
STUDENTAS

Studento#	***ProfVardPav***	**StudentoVardPav**	**Adresas**
4441289	Mykolas	Ona	Gedimino g. 3
4485621	Mykolas	Vilius	Guobų g. 10
4471321	Tomas	Albertas	Parko g. 13

2.1.4 „Daug su daug“ (M:M) asociacijų praktinis įgyvendinimas

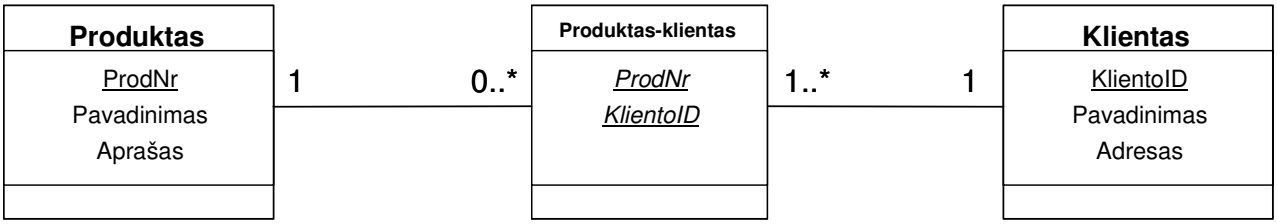
„Daug su daug“ ryšių neįmanoma tiesiogiai atvaizduoti ryšiais kaip 1:1 ir 1:M ryšių. Yra nuomonių, kad M:M ryšių fizinėse lentelėse iš viso nėra – tai tik klasių diagramos koncepcija, o iš tiesų yra du 1:M ryšiai. Kad atvaizduotume M:M asociaciją, reikia sukurti naują ryšį (lentelę), kuris vaizduotų patį ryšį.

Pavyzdžiui, įsivaizduokite produkto-kliento asociaciją (2-4 pav.). Šia asociacija bandoma pavaizduoti, kokie klientai kokius produktus pirko. Aišku, kad tą patį produktą gali pirkti daug klientų, o tas pats klientas gali pirkti įvairius produktus.



2-4 pav. Produkto-kliento asociacijos modelis

Norint įgyvendinti šią asociaciją kaip ryšius, reikia ją perrašyti kaip dvi 1:M asociacijas, kuriose būtų nauja klasė, vaizduojanti patį ryšį. Rezultatas pavaizduotas 2-5 pav.



2-5 pav. Produkto-kliento asociacija, perrašyta kaip dvi 1:M asociacijos

Abu „1“ pusės lentelių raktai (ProdNr ir KlientoID) įkeliami į ryšių „M“ pusę ir įrašom į naują produktų-klientų lentelę kaip išoriniai raktai. Produktų-klientų lentelės raktas yra (ProdNr, KlientoID), nes eilutei nustatyti reikia žinoti abi reikšmes. Abu produktų-klientų lentelės laukai yra sudėtinio rakto dalys, ir abu yra išoriniai raktai. Ši nauja lentelė vadinama sankirtos lentele, kryžminių nuorodų lentele arba asociacijų lentele – kaip kam labiau patinka. Mes vartosime terminą „**asociacijų lentelė**“, nes jis atitinka UML terminą „asociacija“.

Gauti ryšiai atrodys taip:

Produktas(ProdNr, Pavadinimas, Aprašas)
Produktas-klientas(*ProdNr*, *KlientoID*)
Klientas(KlientoID, Pavadinimas, Adresas)

Fizinės lentelės atrodys kaip parodyta 2-6 pav.

ProdNr	Pavadinimas	Aprašas
10035	Sėdynės apmušalas	Plaunamas
26658	Oro gaiviklis	Pušis
6685	Padanga	14R42
35891	Domkratas	2 tonos

ProdNr	*KlientoID*
10035	1
10035	2
26658	2
26658	3
6685	1
6685	2

KlientoID	Pavadinimas	Adresas
1	D. Jonaitis	Gedimino g. 3
2	T. Blinda	Kaštonų g. 8-2
3	V. Alekna	Parko g. 13

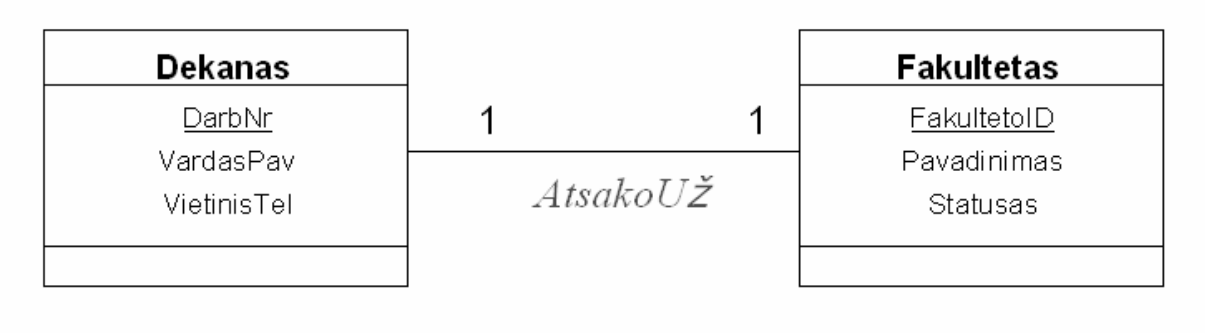
2-6 pav. Praktiškai įgyvendinta produkto-kliento asociacija

2.1.5 Pratimai

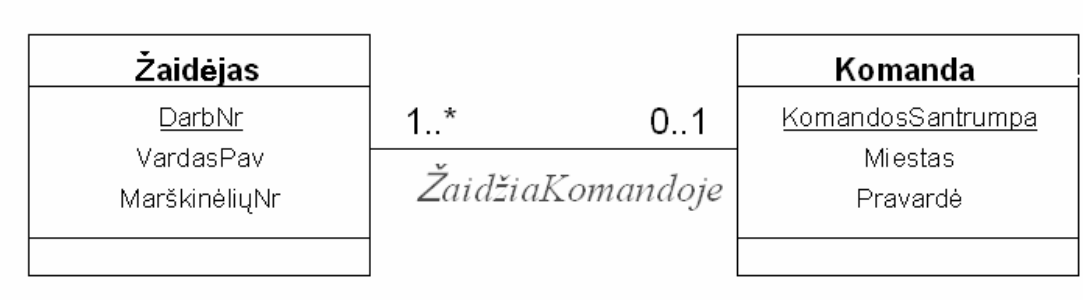
Pabandykite patys įgyvendinti kelias asociacijas. Kad galėtumėte pasitikrinti savo atsakymus, atsakymai pateikti iš karto po užduočių.

Raktus pabraukite, o išorinius raktus rašykite *kursyvu*. Atsakymų forma: ryšis (1laukas, 2laukas, …)

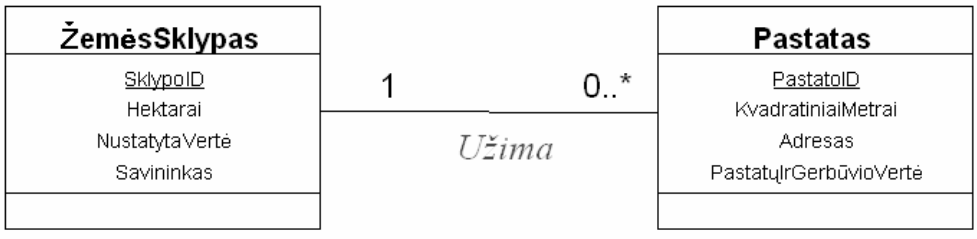
1. Dekanas-fakultetas



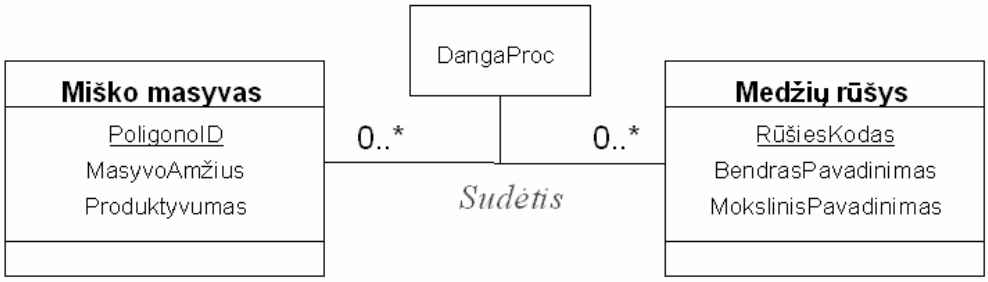
2. Žaidėjas-komanda



3. Žemės sklypas-pastatas



4. Miško masyvas (sklypas)-medžių rūšys



Pateiktų užduočių sprendimai

1. Dekanas-fakultetas

Tai 1:1 ryšys, todėl jį įgyvendinti galima dviem būdais:

Dekanas(DarbNr, *FakultetoID*, VardasPav, VetinisTelefonas)
Fakultetas(FakultetoID, Pavadinimas, Būsena)

ARBA

Dekanas(DarbNr, VardasPav, VietinisTelefonas)
Fakultetas(FakultetoID, *DarbNr*, Pavadinimas, Būsena)

2. Žaidėjas-komanda

Tai M:1 asociacija – komandoje daug žaidėjų, o žaidėjas žaidžia vienoje komandoje. Išorinis raktas pridedamas prie ryšio M pusės. Tik vienas teisingas sprendimas.

Žaidėjas(DarbNr, *KomandosSantrumpa*, VardasPav, MarškinėliųNr)
Komanda(KomandosSantrumpa, Miestas, Pravardė)

3. Žemės sklypas-pastatas

1:M ryšys – sklype gali būti keli pastatai, bet pastatas privalo būti viename sklype. Išorinis raktas pridedamas prie M pusės.

ŽemėsSklypas(SklypoID, Hektarai, NustatytaVertė, Savininkas)
Pastatas(PastatoID, *SklypoID*, KvadratiniaiMetrai, Adresas, PastatųIrGerbūvioVertė)

4. Miško masyvas(sklypas)-medžių rūšys

M:M ryšys, be to, turi asociacijos klasę (atributai priskirti pačiam ryšiui atributų). Asociacijų klasės visada priklauso M:M ryšiams, taigi šis klausimas yra proga susipažinti ir su M:M ryšiais, ir su asociacijų klasėmis.

Reikia sukurti naują klasę ir paversti šį ryšį dviem 1:M asociacijomis. Atributas *DangaProc* bus priskirtas šiai naujai klasei.

MiškoMasyvas(PoligonoID, MasyvoAmžius, Produktyvumas)
Masyvas-Rūšys(*PoligonoID*, *Rūšies kodas*, DangaProc)
MedžiųRūšys(RūšiesKodas, BendrasPavadinimas, MokslinisPavadinimas)

2.2 ESRI duomenų struktūros

Pereinant nuo teorijos prie praktikos, reikia aptarti galimas būsimos GIS duomenų struktūras. Kad galėtume sukurti duomenų bazę, reikia išsiaiškinti, kaip duomenų bazė veikia ir išmokti specifinę terminiją. Tada galėsime išmokti klasių diagramos objektus ir asociacijas paversti veikiančia geoduomenų baze.

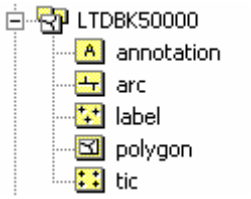
Kiekviena GIS turi savo duomenų formatus. Šiame kurse jūs daug dirbsite su *ArcMap*, todėl svarbu suprasti šios programos erdvinių duomenų terminus ir formatus. Nors kurse pagrindinis dėmesys skiriamas šiai duomenų bazei, vis dar naudojami ir kiti duomenų formatai, pavyzdžiui, *coverage* sandaros sluoksnis, todėl aptarsime ir juos, tik ne taip išsamiai. Šioje temoje apžvelgsime įvairius erdvinių duomenų tipus ir formatus, kuriais jie saugomi.

2.2.1 *Coverage* sandaros sluoksniai

***Coverage* sandaros sluoksniai** – tai vektorinis duomenų formatas, apie 1980 m. sukurtas kaip pagrindinė *Arc/INFO* duomenų struktūra. Šis formatas – tai dvigubos architektūros, arba georeliacinė, struktūra, kurioje saugoma atributų, erdvinė ir topologinė informacija. Erdviniai duomenys saugomi dvejetainiuose failuose, atributai – INFO duomenų bazės programos lentelėje.

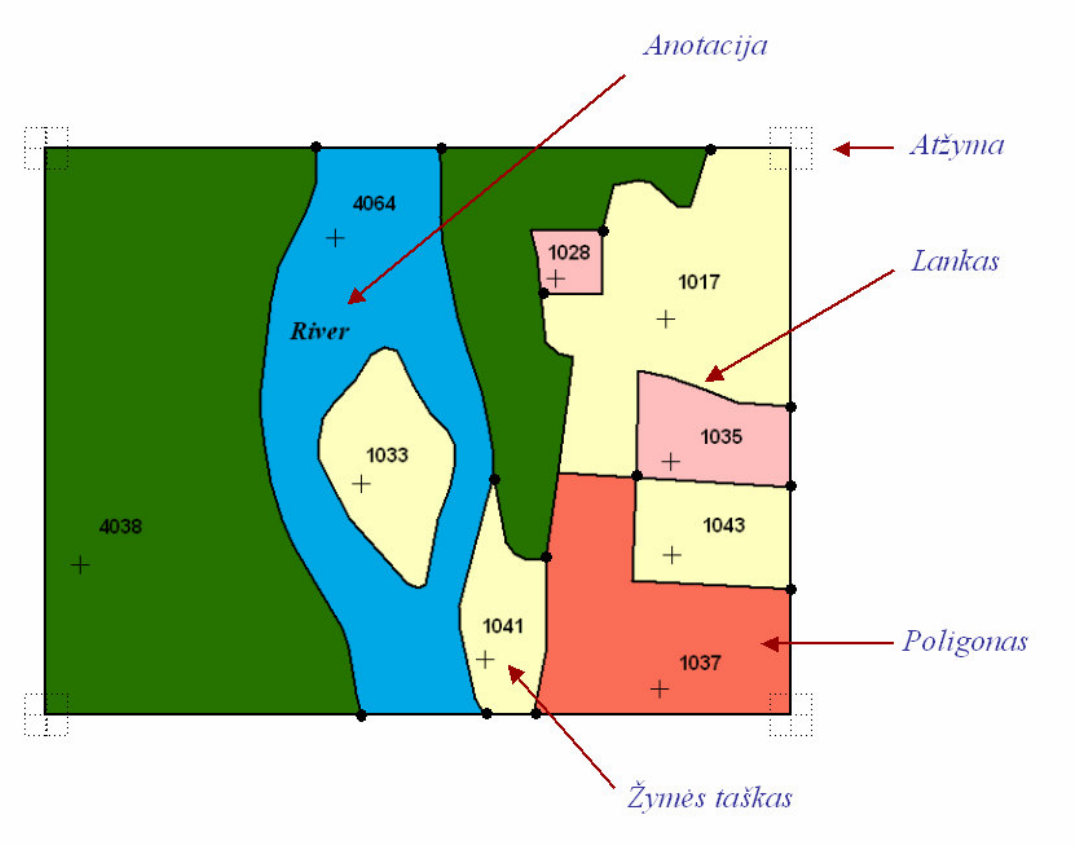
Coverage sluoksnis yra atskiras geografinių objektų, pavyzdžiui, kelių, sklypų, dirvožemio vienetų ar miško masyvų rinkinys. Jis yra daugiaklasis, arba polimorfinis – jame gali būti kelios skirtingos su tuo pačiu *coverage* sluoksniu susijusios **elementų klasės.** Pavyzdžiui, žemės paskirties *coverage* poligonų sluoksnyje gali būti poligonai, jų perimetrus sudarančios linijos, šias linijas nurodantys arba sankirtos taškai ir panaši informacija.

ArcCatalog programa galima pamatyti, kokios elementų klasės yra *coverage* sluoksnyje, o į *ArcMap* duomenų langą galima įkelti atskiras *coverage* sluoksnio klases.
2-7 pav. pavaizduoti Lietuvos 1:50 000 mastelio žemėlapio (LTDBK50000) žemės paskirties poligonai realizuoti *coverage sluoksnio* poligoninėje struktūroje. Kiekvieną elementų klasę aptarsime atskirai.



2-7 pav. Taip poligonų *coverage* sluoksnis atrodo *ArcCatalog* programoje

Šios elementų klasės atitinka įvairias poligonų *coverage* sluoksnio dalis. 2-8 pav. pavaizduota, kaip atrodo žemės paskirties sluoksnio elementų klasės grafinėje išraiškoje.



2-8 pav. Taip poligonų *coverage* sluoksnis atrodo *ArcMap* programoje

Skirtingų *coverage* sluoksnių elementų klasės skirsis atsižvelgiant į tai, ką *coverage* sluoksnis turi vaizduoti. Pavyzdžiui, jei *coverage* sluoksnis skirtas saugoti geoobjektus išreikštus linijomis, t.y. upėms ar keliams, *coverage* sluoksnyje poligonų elementų klasės nebus. 2-9 ir 2-10 pav. pavaizduota, kokios elementų klasės yra atitinkamai linijiniuose ir taškiniuose *coverage* sluoksniuose.

2-9 pav. Linijinis *coverage* sluoksnis (miestų ribos)

2-10 pav. Taškinis *coverage* sluoksnis (miestai)

Coverage sluoksnyje gali būti ir kitų elementų klasių. Visos galimos *coverage* sluoksnio elementų klasės išvardytos ir trumpai paaiškintos 2-3 lentelėje.

2-3 lentelė. *Coverage* sluoksnio elementų klasės

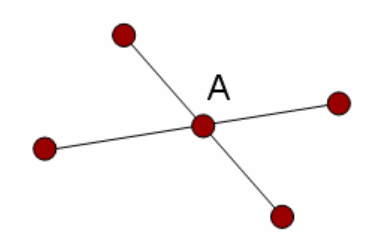
Elementų klasė	Paskirtis
Taškas (point)	x, y koordinatėmis nurodytas taškas, vaizduojantis taškinį geoobjektą
Lankas (arc)	x, y koordinačių rinkinys, vaizduojantis linijinius geoobjektus ar poligonų ribas
Mazgas (node)	Galiniai lanko taškai arba dviejų ar daugiau lankų sujungimo taškas
Poligonas (polygon)	Apribota sritis, atitinkanti poligoninį geoobjektą
Anotacija (annotation)	Teksto eilutė ant žemėlapio, pavyzdžiui, geografinis pavadinimas
Atžyma (tic)	*Coverage* sluoksnio geografiniam orientavimui skirtas taškas. Atžymos skirtos priverstinai orientuoti coverage sluoksnį geografinėje erdvėje
Maršrutas (route)	Linijinis elementas, sudarytas iš vieno ar daugiau lankų arba jų dalių. Maršrutai leidžia sudaryti sudėtingus linijinius elementus, pavyzdžiui, visus vienos upės sistemos intakus galima sujungti į vieną maršrutą
Ryšys (link)	Taškų transformacijų seka, naudojama sluoksniui ištampyti
Regionas (region)	Poligoninis elementas, sudarytas iš vieno ar kelių poligonų. Regionai leidžia sudaryti sudėtingus poligoninius elementus, pavyzdžiui, daugiadalius poligonus (pvz., Indonezija) ir persidengiančius poligonus

Coverage sluoksnį sudaro keli failai, saugomi duomenų kaupiklyje, pavyzdžiui, kietajame diske. Kiekvienas sluoksnis turi atskirą aplanką, kurio pavadinimas toks pat, kaip sluoksnio, ir papildomą aplanką pavadinimu „*info*“, kuriame saugomi visi į vieną aplanką įrašytų sluoksnių INFO duomenų bazės failai. Aplankas, į kurį įrašyti vienas ar keli sluoksniai, vadinamas **darbo sritimi (*workspace*)**; *info* aplankas bendras visiems tos pačios duomenų srities sluoksniams. Kadangi sluoksnį sudarantys failai įrašyti į daugiau negu vieną duomenų srities aplanką, naudotojams nepatartume kopijuoti ar perkelti sluoksnių operacinės sistemos priemonėmis, pavyzdžiui, *Windows Explorer*, kad nesugadintų duomenų. Sluoksnius perkelkite, kopijuokite ar keiskite *ArcCatalog* programa.

Ankstesniuose kursuose tikriausiai išmokote, kad **topologija** nurodo erdvinius ryšius tarp susijungusių arba gretimų geografinių duomenų elementų. *Coverage* sluoksnyje topologiniai ryšiai saugomi pačiame coverage sluoksnio faile. Pavyzdžiui, sluoksnyje

saugoma topologinė informacija apie tai, kaip sujungtos linijos, kurios linijos sudaro konkrečius poligonus ir kurie poligonai yra gretimi. Būtent todėl įvairius geoobjektus sudarančios elementų klasės (pvz., taškai, linijos, poligonai) sluoksnyje saugomos kaip atskiros dalys. Jei yra sukurti ir įrašyti topologiniai ryšiai, paprasčiau atlikti įprastines analizės funkcijas, pavyzdžiui, modeliuoti srautą per sujungtas tinklo linijas, jungti gretimus panašių charakteristikų poligonus arba persidengiančius geografinius elementus. Nustačius duomenų topologinius ryšius ir išsaugojus išreikštu pavidalu, tokie analizės veiksmai atliekami daug greičiau.

Tačiau šiuo labai struktūruotų duomenų saugojimo formatu sunkiau pavaizduoti tam tikrus geografinius reiškinius. Kad būtų išsaugota linijinių elementų jungimosi informacija, kiekvienas linijos segmentas tarp sankirtos su kitomis linijomis taškų į coverage sluoksnį įrašomas kaip atskiras lankas. Pavyzdžiui, įsivaizduokite du linijinius elementus, išdėstytus taip, kaip pavaizduota 2-11 pav. Net jei operatorius šias linijas įvedė kaip dvi tieses (kaip rašydamas „X“), bet jos kertasi, todėl bus sukurta ir įrašyta į sluoksnį linijų sankirtos taške A esanti viršūnė. Taigi bus gauti keturi paprasti linijiniai elementai, turintys bendrą tašką A.



2-11 pav. Susikertantys lankai

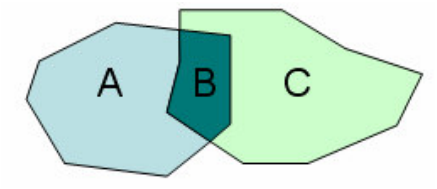
Tam tikrais atvejais tai naudinga, pavyzdžiui, jei vaizduojamos požeminės komunikacijos – kanalizacijos vamzdynai ar pan. Kitais atvejais, pavyzdžiui, vaizduojant gatvių tinklą, gali kilti sunkumų. Gali būti, kad organizacija nori atvaizduoti visus tam tikrą gatvę sudarančius linijų segmentus vienu linijiniu elementu. Tada „gatvės“ elementą reikia sudaryti iš tarp sankirtos taškų esančių segmentų sekos. Linijos turi likti suskaidytos taškuose, kuriuose kertasi su kitais keliais (kad galėtume atlikti tinklo jungimosi analizę), bet mes norime, kad būtų galima kelis lankus valdyti kaip vieną elementą. Tokius elementus vadinsime **daugiadaliais elementais (Multi-Part Feature)**.

Daugiadaliams elementams saugoti sluoksniuose yra maršrutų elementų klasė, leidžianti naudotojui sudaryti naujus elementus iš vieno ar kelių paprastų lankų. Šie lankai gali jungtis į vieną linijinį kelią, eilę sujungtų išsišakojančių kelių arba visai nesijungti. Naudotojams šis linijinių elementų loginis apjungimas gali būti pakankamai didelis papildomas darbas.

Į sluoksnio struktūrą įrašant poligonus taip pat kyla sunkumų. Ir šiuo atveju, kadangi topologija skaičiuojama ir įrašoma į duomenų struktūrą, kiekviena linijomis apribota sritis laikoma atskiru poligonu. Tai gali tikti daugumai taikymo sričių, tačiau kartais tokia struktūra neadekvati.

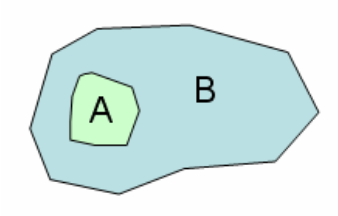
Įsivaizduokite du erdvėje persidengiančius poligonus (2-12 pav. Įskaitmeninus šiuos poligonus, į sluoksnį jie gali būti įrašyti tik kaip trys poligonai. Perdangos sritis (pažymėta B raide) turi tapti nauju poligonu, todėl tikros A ir C poligonų perdangos nebelieka. Šie

poligonai gali vaizduoti dviejų lokių individualias teritorijos, kurių maža dalimi naudojasi abu lokiai. Sluoksnyje paprastais poligonais to atvaizduoti neįmanoma.



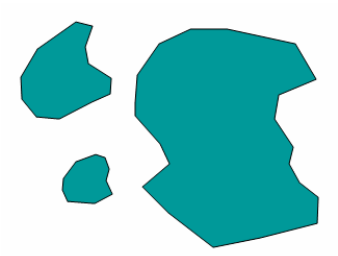
2-12 pav. Persidengiantys poligonai

Kita poligonų *coverage* sluoksnyje problema ta, kad neįmanoma atvaizduoti tuščiavidurių poligonų – reikšmių neturinčių sričių didesniuose poligonuose. Pavyzdžiui, 2-13 pav. pavaizduotas bandymas išreikšti vandens geoobjektą. B poligonas gali vaizduoti ežerą su sala (A poligonas), taigi reikia atvaizduoti, kad B sritis yra vandens elementas, o A sritis nėra vandens elementas. Sluoksnyje A tampa atskiru poligonu, ir nurodyti, kad A „ne vanduo", galima tik atributais. Todėl nepatyrę sluoksnių naudotojai kartais daro klaidų. Pavyzdžiui, skaičiuodami bendrą *coverage* sluoksnio vandens elementų plotą tiesiog sudeda visus – ir tuščius – vandens poligonus, todėl gauna per didelį vandens plotą.



2-13 pav. Tuščiaviduris poligonas

Kaip ir linijiniai elementai, poligonai taip pat gali būti daugiadaliai. 2-14 pav. pavaizduoti trys atskiri poligonai. Jei jie vaizduoja vieną valstybę sudarančias salas, pavyzdžiui, Jungtinę karalystę arba Indoneziją, turi būti galimybė sujungti juos į vieną daugiadalį poligoną.



2-14 pav. Daugiadalis poligonas

Sluoksniuose sudėtingų poligonų problemoms spręsti (perdangoms, tuštumoms ir daugiadaliams poligonams formuoti) skirta regionų elementų klasė. Kaip ir maršrutai, regionai leidžia jungti poligonus į logines grupes įvairiais tikslais, pavyzdžiui, kaip ką tik aptartu atveju. Kaip ir maršrutams, regionams kurti ir tvarkyti reikia daugiau darbo.

Kai buvo kuriamas sluoksnių formatas, programinės įrangos (pvz., operacinių sistemų ir INFO DBVS) galimybės buvo ribotos, todėl failų ir aplankų pavadinimams galioja tam tikri

papildomi apribojimai. Dirbantiems su *coverage* sluoksniais naudotojams derėtų juos prisiminti.

- Dėl INFO DBMS apribojimų sluoksnių pavadinimai turi būti ne ilgesni negu 13 simbolių ir gali būti sudaryti tik iš raidžių, skaičių, brūkšnelių ir pabraukimo brūkšnių. Kitų simbolių pavadinime būti negali.
- Niekur darbo srities kelyje ir sluoksnio pavadinime negali būti tarpų. Tai reiškia, kad, pavyzdžiui, negalima įrašyti tarpų kelyje iki *coverage* sluoksnio. Jei kelias bus C:\Lab 6 Duomenys\Ezerai (su tarpais), *ArcMap* gali neteisingai perskaityti sluoksnį.

Nors dabar ESRI kaip duomenų struktūrą naudoja geoduomenų bazę, *coverage* sluoksniai vis dar naudojami ir failai šiuo formatu vis dar platinami. Taigi studentai turėtų pakankamai gerai žinoti šią struktūrą, kad prireikus galėtų ją apdoroti ir konvertuoti. Geoduomenų bazės formatą aptarsime šioje kurso dalyje vėliau. Kadangi naujesnės ESRI produktų linijos (*ArcGIS*) pagrindas yra geoduomenų bazė, ESRI nebepalaiko *coverage* sluoksnių formato. *ArcGIS* programų naudotojai šio formato failus gali importuoti, skaityti arba eksportuoti, bet tiesiogiai redaguoti *coverage* sluoksnių nejmanoma. Norint *coverage* sluoksnius redaguoti, juos reikia eksportuoti į geoduomenų bazės ar SHAPE formatą arba naudotis *ArcEdit* programa.

2.2.2 SHAPE failas

Nors *coverage* sluoksniuose griežtai kontroliuojama erdvinių duomenų topologinė struktūra ir todėl galima greitai atlikti įvairią analizę, dėl modelio sudėtingumo paprastos operacijos (pvz. duomenų peržiūra) yra palyginti lėtos. SHAPE failų formatas buvo pristatytas dešimtojo dešimtmečio viduryje kartu su *ArcView 2.0* programa – tai buvo bandymas sukurti paprastą, greitai skaitomą ir rodomą formatą.

Kaip ir sluoksniai, SHAPE failas yra vektorinis ir georeliacinis formatas. Erdviniai duomenys saugomi nuosavu dvejetainiu formatu, o atributai – *dBase* duomenų bazės formatu. Viename SHAPE formato faile saugomi vieno tipo elementai (taškai, linijos ar poligonai). Šiuo metu SHAPE failai tikriausiai yra populiariausias erdvinių duomenų bazės formatas, nes formato specifikacija prieinama ir yra kelios nemokamos arba nebrangios programos, galinčios šiuos failus skaityti. Be to, skaityti ir importuoti šiuos failus gali dauguma komercinių GIS programų.

Kad erdvinius duomenis saugoti paprasčiau, SHAPE failuose topologiniai ryšiai nesaugomi ir priverstinai nesudaromi. Todėl užimamos vietos atžvilgiu tai nėra pats efektyviausias būdas – pavyzdžiui, kartojamos bendros poligonų ribos, bet erdviniai elementai nurodomi yra savarankiškais koordinačių rinkiniais. Taigi norint parodyti tam tikrą tašką, liniją ar poligoną nereikia skaityti informaciją iš kelių lentelių ar įrašų. Kadangi SHAPE failuose topologiniai ryšiai nesaugomi, neįmanoma užtikrinti erdvinio vientisumo, pavyzdžiui, garantuoti, kad kadastro sklypų duomenų bazėje nebus tarpų ar perdangų. Topologinius ryšius, pavyzdžiui, bendras linijas ar bendrus taškus, galima priverstinai įjungti redagavimo seansuose programos lygmeniu – tam skirtos *ArcMap* žemėlapio topologijos (**Map Topology**) funkcija ir specialūs topologiniai redagavimo įrankiai, bet gauti erdviniai ryšiai į patį SHAPE failą neįrašomi.

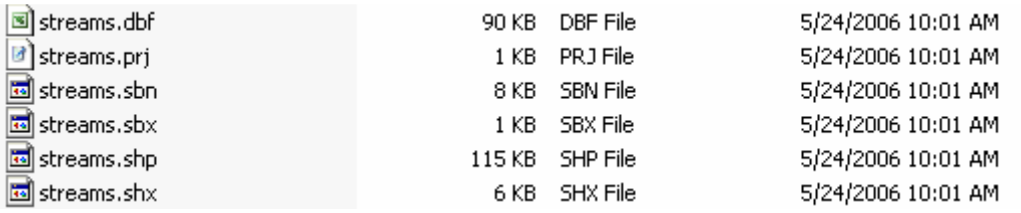
SHAPE failą sudaro į duomenų laikmeną įrašyta failų grupė. SHAPE failas, pavyzdžiui, kelių rinkinys, yra ne vienas fizinis failas, kaip atrodytų iš pavadinimo, o keli atskiri failai. Visų jų pavadinimas tas pats, o plėtiniai skiriasi. Trys pagrindiniai failai yra:

SHP: dvejetainis failas su erdvinių duomenų x, y koordinatėmis.

DBF: *dBase* failas, kuriame saugomi erdvinių objektų atributai. DBF yra įprastas failų formatas, šiuos failus galima atverti *dBase*, *Excel* ir kitomis programomis. Atributai su erdviniais objektais susieti 1:1 ryšiu.

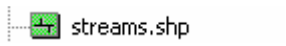
SHX: indekso failas susijusių SHP ir DBF eilučių paieškai pagreitinti.

Taigi kelių rinkinio SHAPE failą sudarys bent trys failai: Keliai.SHP, Keliai.DBF ir Keliai.SHX. Priklausomai nuo to, kaip SHAPE failas buvo naudotas, gali būti dar 3 failai. Jei SHAPE failas turi erdvinį indeksą, bus dar du failai: SBN ir SBX. Jei SHAPE failui nurodyta projekcija, bus ir PRJ failas. SHAPE failą su visais šiais elementais sudarys šeši failai – visi to paties pavadinimo. Pavyzdžiui, 2-15 pav. pavaizduota, kaip upių linijų SHAPE failas gali atrodyti *Windows Explorer* lange.



2-15 pav. Upių SHAPE failas *Windows Explorer* lange

Kadangi SHAPE failai saugomi atskirai viename aplanke, o ne keliuose, juos galima perkelti ar kopijuoti operacinės sistemos priemonėmis, pavyzdžiui, *Windows Explorer*. Tačiau būtinai reikia nukopijuoti visus to paties pavadinimo failus, priešingu atveju galima sugadinti SHAPE failą. Jei naudojamas ESRI produktas, pavyzdžiui, *ArcCatalog*, programa pati pasirūpina, kad visi vieno SHAPE failo failai būtų kartu. Pavyzdžiui, tas pats SHAPE failas *ArcCatalog* lange atrodo tiesiog kaip vienas objektas (2-16 pav.).



2-16 pav. Upių SHAPE failas *ArcCatalog* programoje

Šis formatas buvo sukurtas tada, kai GIS programos buvo pradėtos diegti asmeniniuose kompiuteriuose. Daugelis to meto operacinių sistemų bei *dBase* failų formato apribojimų galioja ir SHAPE failų struktūrai bei pavadinimams. Pavyzdžiui:

- failų pavadinimai turi būti ne ilgesni negu 8 simbolių; pavadinimas turi prasidėti tekstiniu simboliu (A-Z, 0-9); Vieninteliai leidžiami netekstiniai simboliai – pabraukimo brūkšniai (_) ir brūkšneliai (-);
- atributinų laukų pavadinimai turi būti ne ilgesni negu 10 simbolių; kaip ir failų pavadinimuose, vieninteliai leidžiami netekstiniai simboliai – pabraukimo brūkšniai ir brūkšneliai;

- didžiausias tekstinių laukų ilgis – 255 simboliai – labai ilgi tekstiniai aprašai negalimi.

SHAPE failų formatas sukurtas labai paprastas, kad būtų galima greitai peržiūrėti ir redaguoti. Tačiau gali kilti įvairių sunkumų dėl erdvinių duomenų struktūros trūkumo. Šis teiginys nepagrįstas empiriniais faktais, bet ilgametė asmeninė darbo su coverage sluoksnių ir SHAPE failais patirtis rodo, kad SHAPE failus ar jų darbines charakteristikas lengviau sugadinti, ypač dirbant su dideliais, sudėtingais duomenų rinkiniais. Jei dirbate su šiuolaikine *ArcGIS* programine įranga, tokiais atvejais geriau naudoti geoduomenų bazės formatą.

2.2.3 Geoduomenų bazė

Geoduomenų bazė yra naujausia ESRI programų naudojama struktūra, ir šiuo metu ji aktyviai naudojama bei tobulinama. Geoduomenų bazė – tai elementų duomenų rinkinių (taškų, linijų ir poligonų), lentelių, rastrų ir NTT (netaisyklingųjų trikampių tinklas) sankaupa. Visi erdviniai, atributų, lentelių ir topologiniai duomenys saugomi vienoje reliacinėje duomenų bazėje. Viena iš patrauklių geoduomenų bazės savybių yra ta, kad, priešingai negu ankstesni formatai, geoduomenų bazė nesusijusi su konkrečia duomenų bazės programa, pavyzdžiui, INFO arba *dBase*. Geoduomenų bazę galima įdiegti standartinėmis RDBVS programomis, pavyzdžiui, *Access*, DB2, *Informix*, *Oracle* ar *SQL Server*.

Anksčiau aptarėme objektinį duomenų modelį, kuriame duomenys ir elgsena įrašoma į duomenų bazės objektus. Geoduomenų bazėse realizuota daug objektinio modelio aspektų, įskaitant ir objektų elgsenos sampratą. **Elgsena** vadinama geoduomenų bazės gebėjimas apriboti elementų kūrimo ir priežiūros būdus. Šie apribojimai savo ruožtu užtikrina tam tikrą duomenų kokybės lygį, kurį anksčiau reikėjo palaikyti rankiniu būdu. Štai keli duomenų bazėje kontroliuojamos elgsenos tipų pavyzdžiai:

Domeno apribojimai	Kontroliuoja leistinas elemento atributų reikšmes. Pavyzdžiui, įvedama naujo žemės sklypo paskirties reikšmė privalo turėti prasmę – būti „*gyvenamoji*“, „*pramoninė*“ ar pan.
Topologiniai apribojimai	Kontroliuoja erdvinius elementų ryšius. Pavyzdžiui, galima nurodyti, kad visi pastatai privalo būti žemės sklypo viduje (t. y. negali kirsti sklypo ribos).
Geometriniai apribojimai	Kontroliuoja erdvinę elementų struktūrą. Pavyzdžiui, galima nurodyti, kad pastatų kampai privalo būti statūs (90°).
Ryšių apribojimai	Kontroliuoja elementų tarpusavio ryšius. Pavyzdžiui, pašalinant ar perkeliant vandentiekio vamzdį, kartu pašalinamos ar perkeliamos ir visos su juo susijusios sklendės.

Geoduomenų bazę galima laikyti objektiniu-sąryšiniu sprendimu. Duomenims į diską rašyti, užklausoms ir transakcijoms apdoroti bei saugumui užtikrinti naudojama reliacinė duomenų bazė, o pati GIS programa atlieka daugiau su objektiniais komponentais

susijusias funkcijas, pavyzdžiui, kontroliuoja geografinių objektų elgseną ir tvarko objektų asociacijas bei potipius. Iš esmės informaciją paima ir įrašo RDBVS, o duomenų semantiką tvarko taikomoji programa.

ArcGIS 9.2 versija gali dirbti su trimis geoduomenų bazių tipais:

- personalinėmis geoduomenų bazėmis;
- failinėmis geoduomenų bazėmis;
- *ArcSDE* geoduomenų bazėmis (SDE reiškia *Spatial Database Engine* – erdvinių duomenų variklis).

Personalinės geoduomenų bazės saugomos *MS Access* duomenų bazėje. Šioje struktūroje visos geoduomenų bazės dalys saugomos viename *Access* duomenų bazės faile (MDB faile). Personalinės geoduomenų bazės dydžio riba yra *Access* nustatyta riba – 2 GB, bet ESRI dokumentacijoje rašoma, kad duomenų bazei pasiekus nuo 250 iki 500 MB, pastebimai krenta darbo sparta. Vienu metu skaityti personalinės geoduomenų bazės duomenis gali keli naudotojai, rašyti į ją – tik vienas. Šios bazės palaiko visą geoduomenų bazės modelį (t. y. elgsenas, ryšių klases, rastrų katalogus ir kt.) ir joms nereikia pirkti duomenų bazių programų.

Failinė geoduomenų bazė – nauja struktūra, išplečianti personalinių geoduomenų bazių galimybes. Šios bazės irgi yra plačiai paplitęs, nemokamas ir paprastas sprendimas, tinkantis daugybei taikomųjų geoduomenų bazių programų. Be to, joms negalioja *Access* apribojimai (failinių geoduomenų bazių riba yra 1 Terabaitas), o sparta yra didesnė. Kaip ir personalinių duomenų bazių, failinių duomenų bazių naudotojui nereikia investuoti į RDBVS programinę įrangą. Duomenys saugomi keliuose failuose viename duomenų kaupiklio aplanke.

Tos pačios failinės duomenų bazės informaciją skaityti taip pat gali keli naudotojai vienu metu, o galimybės keliems vartotojams vienu metu rašyti į duomenų bazę yra truputį geresnės. Vienu metu tik vienas žmogus gali redaguoti vieną *elementų duomenų rinkinį*, o ne *visą duomenų bazę*, kaip personalinėse geoduomenų bazėse. Pavyzdžiui, vienas žmogus gali redaguoti tos pačios failinės duomenų bazės kelius, kitas – sklypų ribas.

ArcSDE geoduomenų bazių duomenims saugoti naudojama standartinė RDBVS, pavyzdžiui, DB2, *Oracle*, *Informix* ar *SQL Server*. Jas daugiausiai naudoja didesnės organizacijos su dideliu naudotojų skaičiumi, galinčios išnaudoti šių RDBVS architektūros teikiamus duomenų apsaugos, vientisumo ir atsarginių kopijų darymo pranašumus. *ArcSDE* geoduomenų bazės gali būti labai didelės ir vienu metu gali dirbti daug naudotojų.

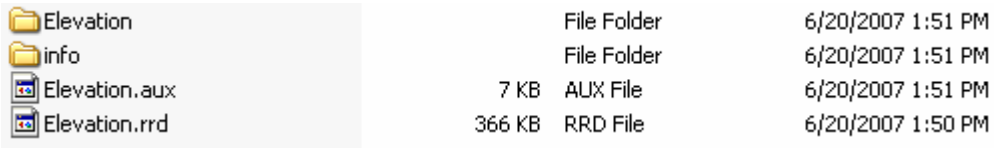
2.2.4 GRID formatas

GRID formatas analogiškas *coverage* sluoksniams, tik skirtas rastriniams duomenims (geografinių matricų duomenims) saugoti. Jis taip pat buvo sukurtas devintajame dešimtmetyje kaip *Arc/INFO* programos dalis. Jame galima sveikų skaičių pavidalu saugoti diskretines atributų reikšmes, arba slankaus kablelio formatu – tolydžių atributų reikšmes. Sveikųjų skaičių tinkleliuose gali būti ir neskaitinėms, pavyzdžiui, žemės paskirties ar

dirvožemio tipo reikšmėms saugoti skirtos reikšmių-atributų lentelės (*Value Attribute Table* – VAT).

GRID failai saugomi panašiai kaip *coverage* sluoksniai. Tai aplankas su keliais failais, kurio pavadinimas toks pat kaip ir GRID failo. GRID failams taip pat galioja darbo srities sąvoka, taigi aplanke, į kurį įrašytas GRID aplankas, bus ir papildomas *info* aplankas. GRID failas gali turėti ir AUX failą, į kurį įrašoma spalvų lentelė, koordinačių sistema, projekcija ir su GRID turiniu susijusi statistika. Šis failas bus sukurtas atliekant pirmą statistinę užduotį. Taip pat gali būti ir RDD failas (*reduced resolution dataset* – sumažintos skiriamosios gebos duomenų rinkinys) arba piramidė. Piramidžių failuose saugomos sumažintos skiriamosios gebos GRID versijos, žymiai paspartinančios duomenų rodymą ekrane įvairiais masteliais.

2-17 pav. pavaizduota, kaip gali atrodyti aukščių GRID failas *Windows Explorer* naršyklėje.



2-17 pav. Aukščių GRID failas *Windows Explorer* naršyklėje

Kaip ir sluoksnių, GRID failų naudotojams nepatartume kopijuoti ar kelti į kitą vietą operacinės sistemos priemonėmis, pavyzdžiui, *Windows Explorer* naršykle, nes vieno failo informacija saugoma keliuose aplankuose. *ArcCatalog* paima visus šiuos failus, todėl GRID failą kopijuoti ar perkelti derėtų su šia programa. Be to, GRID failams galioja tie patys INFO apribojimai, kaip ir sluoksniams: jų pavadinimai turi būti ne ilgesni negu 13 simbolių, nei pavadinime, nei kelyje negali būti netekstinių simbolių (išimtis – pabraukimo brūkšnys ir brūkšnelis).

2.2.5 Rastrai geoduomenų bazėse

Anksčiau rastriniai duomenys buvo visiškai atskiri nuo susijusių vektorinių duomenų. Atsiradus geoduomenų bazės struktūrai, rastriniai duomenys buvo integruoti į didesnę geoduomenų bazės sistemą. Rastrinių duomenų rinkinius tvarkant geoduomenų bazėje, daugeliu atvejų padidėja darbo sparta. Taip pat galima pasinaudoti RDBVS aplinkos pranašumais, pavyzdžiui, saugumu, daugelio vartotojų palaikymu ir klaidų taisymo galimybėmis.

Rastriniai duomenys geoduomenų bazėje gali būti šių pavidalų:

- **rastrinių duomenų rinkinys** – vientisas, iš kelių dalių sumontuotas rastras;
- **rastrų katalogas** – keli atskiri rastrai, kurie gali persidengti arba būti atskirti erdvėje;
- **rastrinis atributas** – rastras arba vaizdas, įrašytas kaip elemento atributas.

Rastrinių duomenų rinkinys apjungia kelis gretimus rastrus į vieną vientisą rastrą. Sudėtinės dalys turi būti tos pačios skiriamosios gebos, to paties formato ir to paties duomenų tipo. Pridedamas prie duomenų rinkinio rastras gali būti pakeistas – iš persidengiančių gardelių bus sudarytas vienas gardelių rinkinys, o jei skiriamoji geba ar gardelių orientacija skiriasi nuo paskirties rastro, gardelių reikšmės gali būti perskaičiuotos. Kadangi pridedamus prie duomenų rinkinio rastrus gali tekti koreguoti, rastrinių duomenų rinkinio atnaujinimas gali ilgai užtrukti. Tačiau kai rastrinių duomenų rinkinys jau sukurtas, įkelti jį ir parodyti ekrane galima labai greitai. Rastrinių duomenų rinkiniai labai efektyvūs dirbant su labai dideliais rastrais.

Rastrų katalogas iš esmės yra priemonė turimiems rastrams valdyti. Į rastrų katalogą įrašyti rastrai gali būti įvairios skiriamosios gebos, įvairių formatų ir duomenų tipų. Erdvėje šie rastrai gali būti persidengę arba atskirti vienas nuo kito. Rastrų katalogai leidžia daug paprasčiau tvarkytis su dideliu rastrų kiekiu. Vienas rastrų katalogo taikymo pavyzdys yra laikinių rastrų sekų saugojimas. Visi šie vaizdai persidengia erdvėje, bet gauti skirtingais laiko momentais, ir juos daug paprasčiau surūšiuoti bei dirbti su keliais vaizdais vienu metu. Taip naudotojai gali greitai surasti ir peržiūrėti dominančio periodo vaizdus.

Rastriniai katalogai sukuriami greičiau – nereikia montuoti vaizdų ir suderinti persidengiančių sričių, taigi atliekama žymiai mažiau apdorojimo operacijų. Tačiau, jei rastrų daug, katalogai ekrane parodomi lėčiau negu rastrinių duomenų rinkiniai, bet juos galima rodyti kaip esamų vaizdų ar teminių rastrų erdvinę aprėptį nurodančius „rėmelius“. Metaduomenys gali būti ir viso katalogo, ir atskirų sudedamųjų rastrų.

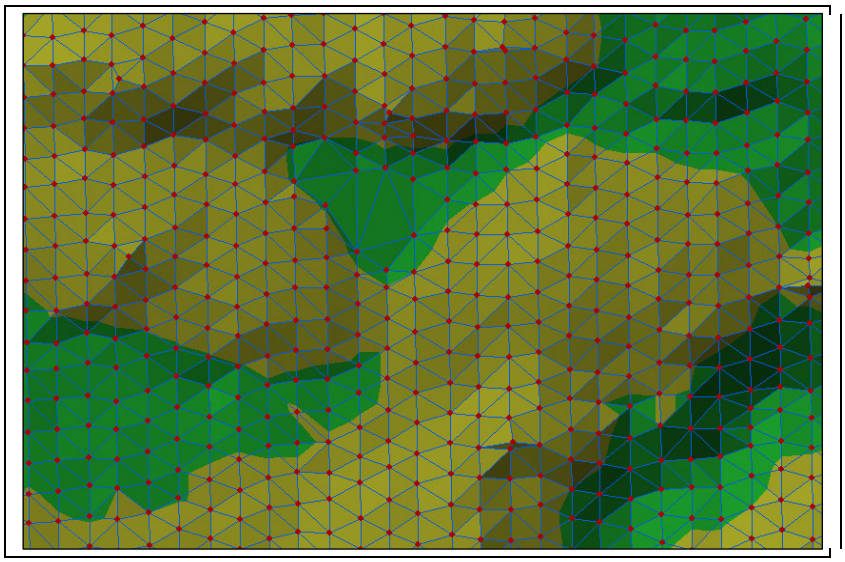
Rastriniai atributai skirti rastrams įrašyti tiesiai į elementų atributų lenteles. Paplitęs šios galimybės pritaikymas – išsaugoti nuotrauką kaip erdvinio elemento atributą. Namo nuotrauką į lentelę galima įrašyti kaip pastato poligono atributą. Prie gatvės segmento galima pridėti gatvės rekonstrukcijos projekto nuotraukas.

Visais atvejais rastriniai duomenys gali būti saugomi arba pačioje geoduomenų bazėje (vadinamieji *valdomi rastrai*), arba pradine savo forma už RDBVS ribų, o duomenų bazėje saugoma tik nuoroda į išorinį rastrą (vadinamieji *nevaldomi rastrai*). Fizinis valdomų rastrų pavidalas priklauso nuo konkrečios geoduomenų bazės tipo. Valdomi personalinės geoduomenų bazės rastrai saugomi aplanke šalia MDB failo standartiniais failų formatais (pvz., ERDAS IMAGINE, JPEG). Failinių geoduomenų bazių valdomi rastrai saugomi failinės geoduomenų bazės aplanke nuosavu formatu. *ArcSDE* duomenų bazėse visi rastrai valdomi ir saugomi pačioje reliacinės duomenų bazės struktūroje.

Kaip ir vektorinių duomenų, rastrų dydžio riba priklauso nuo naudojamos geoduomenų bazės. Personalinėse geoduomenų bazėse ši riba yra 2 GB, failinės geoduomenų bazės rastro dydis neturi viršyti 1 TB, o *ArcSDE* geoduomenų bazės failų dydžio neriboja.

2.2.6 Netaisyklingųjų trikampių tinklai (NTT)

Netaisyklingųjų trikampių tinklas (*Triangulated Irregular Network* – TIN) – tai paviršiams aprašyti skirtas duomenų rinkinys (pavyzdžiui, aukščių reikšmėms nurodyti). 18 pav. pateiktas NTT pavyzdys. Kaip ir *coverage* sluoksniai, NTT yra aplankas su failais, bet jie neturi INFO failų. NTT kataloge yra 7 paviršių aprašantys dvejetainiai failai.



2-18 pav. NTT pavyzdys

NTT sudaro:

Mazgai	Pagal mazgų Z koordinates ir sudaromas NTT. Mazgai atitinka taškinius elementus arba linijinių elementų taškus, iš kurių NTT sudaromas. Jie yra trikampių sankirtos taškai.
Briaunos	Briaunos jungia mazgus su artimiausiais jų kaimynais ir yra NTT trikampių kraštinės.
Trikampiai	Tai TIN plokštumų poligonai. Kadangi žinomos visų trijų trikampio viršūnių koordinatės, galima apskaičiuoti kiekvieno trikampio nuolydį, orientaciją, plotą ir panašias charakteristikas.

Kaip ir *coverage* sluoksniuose, topologinė informacija saugoma NTT failų struktūroje. Apskaičiuojami ir įrašomi kiekvieno trikampio gretimi trikampiai, trikampio kraštinės ir visų trijų kampų mazgai (įskaitant Z koordinates).

2.2.7 Paviršius geoduomenų bazėse

Vis labiau plintant labai didelės skiriamosios gebos technologijoms, pavyzdžiui, *LIDAR* (*Light Detection and Ranging*), statinės skiriamosios gebos išraiška, tokia kaip NTT, dėl didelio duomenų kiekio greitai tapo labai nepatogi dirbant didelėse teritorijose. Darbui su geoduomenų bazėmis buvo sukurta paviršiaus (*terrain*) atvaizdavimo struktūra. Paviršiai yra NTT pagrindu sukurtos kintamos skiriamosios gebos struktūros.

Paviršiuose saugomos kelios elementų klasės, pavyzdžiui, paviršiaus aukščio taškai, lūžio linijos, taip pat rinkinys taisyklių, nurodančių, kaip iš šių elementų sudaryti paviršių. Taip pat apibrėžiamos taisyklės, kaip elementų klases vaizduoti įvairiais masteliais. Pavyzdžiui, vaizduojant reljefą labai smulkiu masteliu nebūtina apdoroti mažų lūžio linijų, kurios neturės pastebimos įtakos reljefo vaizdui. Šiuo atveju smulkios reljefo lūžio linijos bus pradedamos vaizduoti tik nuo tam tikro stambesnio mastelio. Naudojamų paviršiaus taškų

kiekis taip pat priklauso nuo konkrečiam masteliui reikalingo detalumo lygio ir vaizdo kokybės reikalavimų (taigi gaunama daugiau ar mažiau trikampių). Taip iš milijardų paviršiaus taškų sudarytą reljefą galima greitai parodyti ekrane įvairiais masteliais.

Į paviršiaus failą sukurtas paviršiaus atvaizdas neįrašomas. Paviršiaus NTT atvaizdas apskaičiuojamas iš einamajam ekrano masteliui reikalingų elementų. Taigi iš paviršiaus duomenų struktūros gaunamas NTT skaičiuojamas dinamiškai ir niekada nesaugomas visam laikui.

Paviršiaus failų dydžio ribos panašios į kitų duomenų formatų (pvz., vektorinių ar rastrinių duomenų), ir taip pat priklauso nuo naudojamos duomenų bazės tipo. ESRI literatūroje nurodyta, kad personalinėje geoduomenų bazėje paviršiaus riba yra apie 20 milijonų taškų, failinės duomenų bazėje – šimtai milijonų taškų, o *ArcSDE* geoduomenų bazėje jo dydis neribojamas ir priklauso tik nuo laikmenos talpos.

2.3 Geoduomenų bazės komponentai

Ankstesnėse temose mes bendrais bruožais aptarėme, kas yra duomenų bazė, kaip į ją įrašomi ir kaip iš jos paimami duomenys. Šioje temoje panagrinėsime geoduomenų bazės modelio dalis ir jų naudojimą geografinių duomenų elgsenai struktūruoti ir valdyti. Štai pagrindinės geoduomenų bazės dalys, kurias mes išsamiai aptarsime kituose skyriuose:

1. Objektai
2. Objektų klasės
3. Elementai
4. Elementų klasės
5. Elementų duomenų rinkiniai
6. Ryšiai
7. Ryšių klasės
8. Domenai
9. Potipiai
10. Topologija

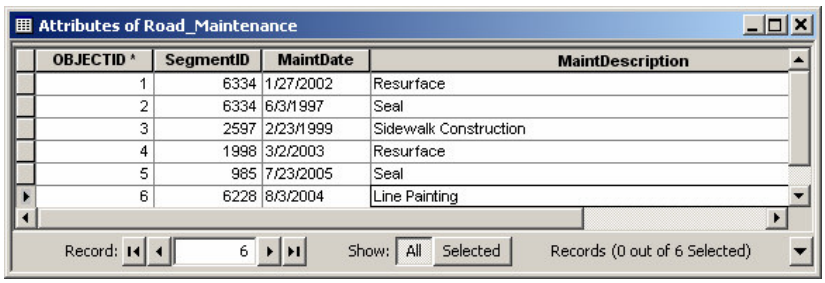
kai kurie kiti specifiniai geoduomenų bazių elementai, pavyzdžiui, rastrinių duomenų rinkiniai, geometriniai tinklai, reljefo duomenų rinkiniai ir kartografiniai atvaizdžiai nagrinėjami susijusiuose kursuose – GII-04, GII-06 ir kt.

2.3.1 Duomenų struktūros komponentai

Objektai ir objektų klasės

Geoduomenų bazės objektų klasės – tai lentelės, kuriose saugomi neerdviniai duomenys (pvz., sklypų savininkai). Tos pačios klasės objektai turi tas pačias savybes bei elgseną. Savybės saugomos lentelėje kaip atributai, o elgsena yra kitas geoduomenų bazės dalis (pvz., domenus arba topologijos taisykles) naudojančių tikrinimo taisyklių rinkinys. Šias dalis aptarsime vėliau. Kaip ir klasių diagramose, objektu vadinamas objektų klasės (pvz., visų sklypų savininkų) atstovas (pvz., Jonas Petraitis). Vieni objektai su kitais gali būti susieti ryšiais.

ArcMap programoje rodoma objektų klasė neturi nei formos (SHAPE) lauko, nei jokio geometrijos tipo atributo, į kurį galima įrašyti erdvinį atvaizdį. 2-19 pav. pavaizduota, kaip atrodo objektų klasės lentelė *ArcMap* aplinkoje.

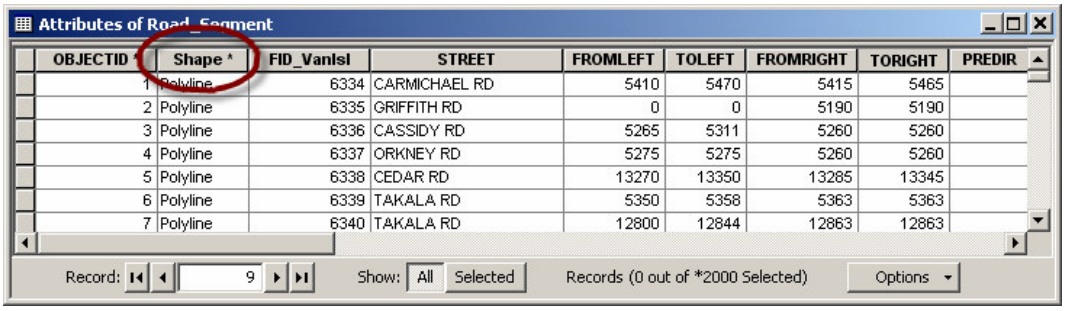


2-19 pav. Objektų klasės lentelės pavyzdys

Elementai ir elementų klasės

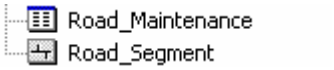
Elementai – tai erdvines charakteristikas turintys objektai (dar vadinami geoobjektais), žemėlapio sluoksnyje vaizduojantys tikrus objektus, pavyzdžiui, žemės sklypų poligonai. Elementų klasės yra to paties geometrijos tipo ir tuos pačius atributus turinčių elementų rinkiniai (pvz., taškai, linijos ar poligonai). Elementų klasė gali būti visi keliai, o jos elementas – viena gatvė. Elementų klases taip pat galima ryšiais susieti su kitomis objektų ar elementų klasėmis. Elementų klasė savo koncepcija primena SHAPE failą.

ArcMap programoje rodoma elementų klasės atributų lentelė turi formos (*Shape*) lauką. Tai reiškia, kad šios lentelės elementai turi erdvinį atvaizdį ir juos galima pavaizduoti žemėlapyje taškiniais, linijiniais ar ploto simboliais. 2-20 pav. pavaizduota, kaip atrodo elementų klasės atributų lentelė *ArcMap* programoje.



2-20 pav. Elementų klasės lentelės pavyzdys

ArcCatalog programoje objektų ir elementų klasės atrodo labai skirtingai. Pavyzdžiui, 2-21 pav. pavaizduota neerdvinių objektų klasė Road_Maintenance (kelių priežiūra) ir linijinių elementų klasė Road_Segment (kelių segmentai). Naudotojai gali lengvai atskirti elementų klases nuo objektų klasių pagal jų piktogramą.



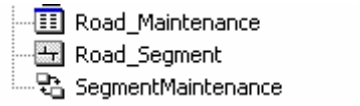
2-21 pav. Elementų ir objektų klasės *ArcCatalog* programoje

Ryšiai ir ryšių klasės

Sąryšiai – tai asociacijos tarp dviejų ar daugiau geoduomenų bazės objektų; objektai gali būti erdviniai (elementai) arba neerdviniai (lentelių eilutės). Ryšiai yra klasių diagramos asociacijų atitikmuo geoduomenų bazėje, jie įgyvendinami ryšių klasėmis. Ryšių klasės sudaro nuolatinę jungtį tarp geoduomenų bazės objektų ar elementų, pavyzdžiui, tarp žemės sklypo ir žemės savininko (-ų).

Kaip ir klasių diagramų asociacijos, ryšiai turi kartotinumą, taigi galima nurodyti ryšio tipą – 1:1, 1:M arba M:M. M:M ryšiams ryšių klasė sukuria naują asociacijos klasę, vaizduojančią patį ryšį. Ryšiai taip pat gali įjungti tam tikrus duomenų bazės veiksmus, pavyzdžiui, pakopinį šalinimą (pavyzdžiui, pašalinamos visos su šalinamu vandentiekio vamzdžiu susietos sklendės), perkėlimą kartu su susijusiu elementu arba naudotojo nustatytą elgseną.

Ryšių klasės geoduomenų bazėje savybės lemia, kokių klasių objektai ir / arba elementai gali sudaryti konkrečią asociaciją bei šios asociacijos pobūdį. 2-22 pav. pavaizduota ryšio klasė, siejanti kelių segmentus su juose vykdomais priežiūros darbais.



2-22 pav. Ryšio klasė *ArcCatalog* programoje

Trys aptarti elementai – objektų klasės, elementų klasės ir ryšių klasės – iš esmės yra pagrindinės UML klasių diagramų sudedamosios dalys. 2-4 lentelėje pateikti pagrindiniai geoduomenų bazių terminai ir juos atitinkantys reliacinių duomenų bazių terminai.

2-4 lentelė. Geoduomenų bazių ir reliacinių duomenų bazių sąvokų palyginimas

Geoduomenų bazės elementas	Reliacinių duomenų bazių atitikmuo
Atributas	Stulpelis, laukas
Objektas	Eilutė
Objektų klasė	Lentelė
Elementas	Eilutė su erdviniais atributais
Elementų klasė	Lentelė su erdviniais atributais
Ryšys	Asociacija tarp dviejų eilučių
Ryšių klasė	Asociacija tarp dviejų lentelių

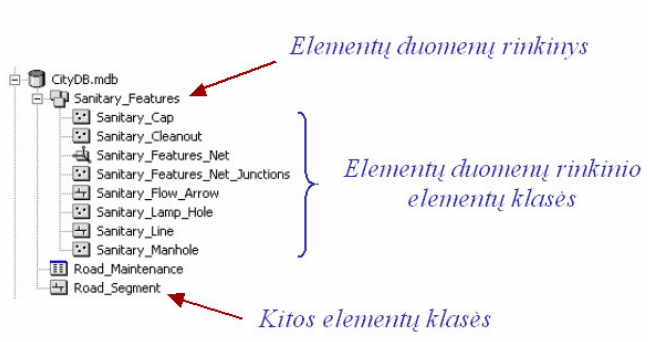
Elementų duomenų rinkiniai

Elementų duomenų rinkiniai yra susijusių elementų klasių „konteineriai“. Juos galima naudoti susijusioms elementų klasėms apjungti į logines grupes. Ryšys tarp klasių gali būti erdvinis arba teminis. Vieno rinkinio elementų klasės privalo turėti tas pačias erdvines charakteristikas (koordinačių sistemą, aprėptį ir pan.).

Elementų rinkiniai būtini geoduomenų bazės reljefo duomenims kurti. Jau išsiaiškinome, kad reljefo duomenų rinkiniuose gali būti kelios elementų klasės, iš kurių sudaromas trimatis NTT atvaizdis. Pavyzdžiui, gali būti aukščio taškus vaizduojančios taškinių elementų klasės ir griežtas arba švelnias lūžių linijas vaizduojančios linijinių elementų klasės. Iš visų šių klasių kartu ir gaunamas reljefas. Kad būtų galima pasinaudoti šia reljefo struktūra, visos elementų klasės turi būti viename elementų rinkinyje. Tas pats principas galioja ir topologijos taisyklėms, tinklo duomenų rinkiniams bei geometriniams tinklams: visos juos sudarančios elementų klasės turi būti viename elementų rinkinyje.

Elementų rinkinius galima naudoti ir duomenų redagavimo teisių kontrolei. Jei nustatomos tam tikros teisės kuriam nors elementų rinkiniui, tas pačias teises įgyja ir visos jo elementų klasės. Šis mechanizmas leidžia geoduomenų bazės administratoriams į duomenų rinkinius apjungti elementų klases, kurias prižiūri skirtingos naudotojų grupės. Taip sutaupoma daug darbo, nes nebereikia prižiūrėti kelių skirtingų prieigos teisių rinkinių.

ArcCatalog programoje elementų duomenų rinkiniai žymimi atskira piktograma, ir matyti, kurios elementų klasės priklauso konkrečiam elementų duomenų rinkiniui. Elementų klasės geoduomenų bazėje gali būti pavienės arba priklausyti elementų duomenų rinkiniui. 2-23 pav. pavaizduota, kaip abu šie atvejai atrodo *ArcCatalog* programoje.



2-23 pav. Elementų duomenų rinkinys

Erdvinės charakteristikos

Erdvinės charakteristikos skirtos geoduomenų bazės objektams susieti su tikromis žemės vietomis. Kad galėtume integruoti visas geoduomenų bazės elementų klases, kiekvienai elementų klasei reikia nurodyti koordinačių sistemą. Erdvines elementų klasės charakteristikas sudaro koordinačių sistema ir erdvinės ribos. Erdvinės ribos yra tiesiog erdvės srities, kurios viduje turi būti visi elementai, x, y ir z koordinatės. Kaip minėjome, elementų duomenų rinkinyje esančių elementų klasių erdvinės charakteristikos turi būti tos pačios.

2.3.2 Elgsenos valdymo priemonės

Geoduomenų bazėje elgsena yra priemonė duomenų kokybei kontroliuoti ir duomenų kūrimo bei redagavimo procesams paspartinti. Tam skirta daug įrankių, kurie labai skiriasi sudėtingumu ir galimybėmis. Paprasčiausias duomenų bazės elementų kontrolės būdas – kontroliuoti jų elementų klasės apibrėžimą, pavyzdžiui, geometrijos tipą (taškas, linija, poligonas) ir galimus atributus. Sudėtingiausias būdas – parašyti programą, kuri kontroliuotų labai specifinius elementų egzistavimo ir ryšių su kitais objektais aspektus. Elgsenos valdymą galima įsivaizduoti kaip seką įvairių priemonių, kurių kiekviena suteikia geoduomenų bazės objektams vis daugiau sumanumo (2-24 pav.).

2-24 pav. Elgsenos galimybių kontinuumas

Potipiai

Potipiai – tai elementų klasių kategorijos, leidžiančios naudotojams nurodyti objektų skirtumus arba suskirstyti juos į kategorijas nekuriant naujų elementų klasių. Pagal prasmę geoduomenų bazių potipių sąvoka atitinka UML klasių diagramų temoje aptartą paveldėjimo sąvoką. Pavyzdžiui, galima sukurti *VandentiekioVamzdžių* elementų klasę, kuri būtų superklasė, ir kelis poklasius, vaizduojančius įvairius vamzdžių tipus, pavyzdžiui, *Betoninius*, *Plieninius* ir *Plastikinius*. Kiekvienas potipis gali turėti savo leistiną skersmenį ar slėgį. Potipiai suteikia galimybę patobulinti vamzdžių tipą nekuriant trijų skirtingų elementų klasių.

Tačiau geoduomenų bazės potipių įgyvendinimo būdas skiriasi nuo UML paveldėjimo asociacijos sudarymo būdo. UML poklasio apibrėžimas leidžia poklasiui paveldėti visus superklasės atributus ir metodus, taip pat apibrėžti papildomus, tik poklasiui priklausančius atributus ir metodus. Anksčiau aptarėme *Paukščių* superklasės pavyzdį – ji turi atributus, nurodančius, kad paukščiai turi plunksnas ir snapą, ir jo *Gulbių* poklasį – jos irgi turi plunksnas ir snapą, ir dar yra baltos bei turi plėvėtas kojas.

Geoduomenų bazėje potipiai negali turėti skirtingai apibrėžtų atributų (stulpelių). Visi elementų klasės potipiai turi tą patį atributų rinkinį, jiems galima tik priskirti kitas išankstines atributų reikšmes arba kitus atributų domenus. *VandentiekioVamzdžių* pavyzdyje visi trys potipiai turėtų tuos pačius *Skersmens* ir *VardinioSlėgio* atributus, bet kiekvienas galėtų turėti skirtingas išankstines reikšmes ir skirtingus domenus, atitinkančius jų konstrukcinių medžiagų charakteristikas. Pavyzdžiui, plieninis vamzdis gali atlaikyti didesnį slėgį ir turėti didesnį skersmenį, negu plastikinis.

Kartais projektuojant reikia priimti sprendimą, ar kurti vieną elementų klasę su dviem potipiais, ar dvi atskiras elementų klases. Tarkime, yra du kelių tipai: viešieji keliai ir privatūs keliai (privatūs gali būti keliai, nutiesti ir prižiūrimi miškininkystės ar kasybos įmonių). Pagrindinis stimulas naudoti poklasius – darbo sparta. Labai apytiksliu vertinimu, geoduomenų bazė su keliomis elementų klasėmis ir keliais jų potipiais veiks žymiai greičiau negu geoduomenų bazė su daug elementų klasių ir be potipių. Jei galėtume atskirti viešuosius kelius nuo privačių vien tik pagal išankstines reikšmes, atributų domenus ir jungimosi taisykles, geriausia būtų sukurti vieną elementų klasę su potipiais.

Jei šių dviejų kelių tipų atributai visiškai skirtingi, arba jei jiems reikia suteikti skirtingas prieigos teises, tada būtina sukurti dvi atskiras elementų klases. Pavyzdžiui, jei viešieji keliai turi adresų diapazonus ir pavadinimus, o privatūs vadinami juos turinčios įmonės pavadinimu ir turi būseną (aktyvūs arba ne), potipių naudoti negalima. Be to, jei kelių elementus įskaitmenina ir juos prižiūri dvi skirtingos naudotojų grupės, turinčios skirtingas prieigos prie duomenų bazės teises, tada taip pat reikia dviejų atskirų elementų klasių.

Elementų klasių potipiai atrodo kaip atskiros elementų klasės. Kuriam poklasiui elementas priklauso, nurodo tam tikras atributas. Išskleidžiamajame potipių sąraše nurodomas visas klasifikacijos pavadinimas, nors iš tiesų geoduomenų bazėje gali būti saugomas skaitmeninis kodas. Taip sutaupoma vietos, nes duomenų bazėje saugomi skaičiai (1, 2, 3 ...), o naudotojui pateikiamos frazės arba kategorijos (*Žvyrkelis*, *KeliasSuKietaDanga*, ...). Dėl to daugeliu atvejų šias reikšmes galima greičiau įvesti iš išskleidžiamojo sąrašo.

Šiame sąraše pasirinkus tam tikrą potipį, bus automatiškai nustatyti visi kiti potipio domenų kontroliuojami atributai ir išankstinės reikšmės. Pavyzdžiui, pasirinkus *Žvyrkelio* potipį, nustatomos 2 eismo juostos ir išankstinė leistino greičio reikšmė – 50 km/h (2-25 pav.).

	OBJECTID*	SHAPE*	SurfaceCode	Lanes	SpeedLimit	SHAPE_Length
▶	4	Polyline	Gravel Road	2	50	1869.582964

2-25 pav. Atributų lentelė su potipiais

Domenai ir tikrinimas

Atributo **domenas** nurodo galimų atributo lauko reikšmių diapazoną. Pavyzdžiui, galima nurodyti, kad vamzdžio skersmuo gali būti bet kokia reikšmė nuo 5 iki 20 cm, arba apriboti žemės paskirties klasifikaciją baigtiniu reikšmių sąrašu. Domenai yra priemonė duomenų bazės kokybei padidinti. Jei iš anksto nustatysime, kokias reikšmes galima įvesti į tam tikrą duomenų bazės lauką, ir tikrinsime, ar įvestos reikšmės atitinka teisingas, galėsime daug labiau pasitikėti savo duomenų bazės duomenimis. Nors negalime garantuoti, kad duomenys bus teisingi, bet galime apsidrausti nuo grubių klaidų. Pavyzdžiui, jei nustatysime 5 – 20 cm intervalinį domeną, vis tiek negalėsime garantuoti, kad operatorius vietoj 10 cm vamzdžio skersmens neįves 12 cm; bet išvengsime, pavyzdžiui, teksto rinkimo klaidų – 100 cm vamzdžio skersmens įvesti nebus įmanoma.

Geoduomenų bazės turi du domenų tipus:

Intervaliniai domenai: nustato leistiną tolydaus kintamojo reikšmių intervalą. Pavyzdžiui, vardinį vamzdžio slėgį galima apriboti reikšmių

intervalu nuo 2,5 iki 7 barų. Anksčiau aprašytas vamzdžių skersmens apribojimas intervalu nuo 5 iki 20 cm taip pat yra intervalinio domeno pavyzdys.

Koduotų reikšmių domenai: nustato leistinų atributo reikšmių sąrašą. Koduotų reikšmių domenus galima naudoti diskretiniams kintamiesiems, pavyzdžiui, žemės paskirties tipui, dirvožemio tipui arba vamzdžių medžiagai. Šiais atvejais leistinų verčių sąrašas baigtinis, ir jį galima sudaryti projekto pradžioje.

Domenai sudaromi ne atskiroms klasėms, o visai duomenų bazei ir turi unikalius pavadinimus. Todėl tuos pačius domenus galima naudoti kelių elementų ar objektų klasių atributams. Įvedant atributų reikšmes, intervaliniai domenai tiesiog perspėja, jei įvedama reikšmė yra už intervalo ribų. Koduotų reikšmių domenai atributų lentelėje rodomi kaip išskleidžiamieji sąrašai.

Domenai tikrina atributų reikšmes, bet jie yra tik viena iš daugelio duomenų tikrinimo priemonių ir metodų. Kitos geoduomenų bazių duomenų tikrinimo priemonės:

Jungimosi taisyklės: leidžia patikrinti, ar teisingi sujungtų tinklo elementų atributai. Pavyzdžiui, galima užtikrinti, kad 10 cm vamzdis nebus tiesiogiai prijungtas prie 15 cm vamzdžio be redukcinės movos.

Ryšių taisyklės: leidžia nustatyti ryšių klasės elementų kartotinumą (1:1, 1:M arba M:M). Be to, galima nustatyti ir specifinį kartotinumą. Pavyzdžiui, prie tam tikro tipo movos būtina prijungti tris vamzdžius, arba reikia nurodyti, kad sklypai gali priklausyti tik vienam arba dviem savininkams. Ryšių taisyklės gali apjungti ir bendrus kartotinumus, pavyzdžiui, 1:M, ir tikslų minimumą ar maksimumą nurodančius duomenų bazės apribojimus, aptartus šio kurso 1 dalyje.

Naudotojo kodas: Specifinėms taikomosioms programoms programuotojas gali parašyti elementų atributų arba struktūros tikrinimo kodą ir pritaikyti ypatingus kriterijus.

Atliekant erdvinį elementų redagavimą, dažnai pasitaiko, kad vienas elementas suskaldomas į du arba du elementai sujungiami į vieną. GIS programa gali lengvai atlikti šioms operacijoms reikalingas erdvines manipuliacijas pati, be naudotojo nurodymų, tačiau šių operacijų įtaka apdorojamų elementų atributams nebūtinai yra žinoma iš anksto. Pavyzdžiui, jei du elementai jungiami į vieną, kaip apskaičiuoti gauto jungtinio elemento atributą? Atributo verčių elgseną skaldant ar jungiant elementus reguliuoja skaldymo taisyklės (**Split Policy**) ir jungimo taisyklės (**Merge Policy**). Kiekvienam geoduomenų bazės domenui galima priskirti ir skaldymo, ir jungimo taisyklę.

Skaldymo taisyklės nurodo, kaip iš vienos skaldomo elemento atributo reikšmės gauti dvi ar daugiau reikšmių. Galimos skaldymo taisyklių nuostatos:

Išankstinė reikšmė: kai esamas elementas dalijamas į naujus elementus, jų atributams priskiriama išankstinė reikšmė, kuri gali būti nesusijusi su pradinio elemento atributo reikšme. Išankstinės

reikšmės nuostata naudojama tada, kai didžiąja dauguma atvejų tinka viena reikšmė. Pavyzdžiui, jei dauguma miesto vandentiekio vamzdynų plieniniai, iš padalinto vamzdžio elemento bus gauti du nauji plieniniai vamzdžiai, nesvarbu, kokia iš tikro yra dalijamo vamzdžio medžiaga.

Kopija: kai esamas elementas dalijamas į naujus elementus, visų jų atributams priskiriama pradinio elemento atributo reikšmė. Šiuo atveju padalinę plieninio vamzdžio elementą į dvi dalis, gausime du naujus plieninių vamzdžių elementus.

Geometrijos santykis: jei atributas skaitinis, pradinio elemento atributo reikšmė dauginama iš naujam elementui tenkančios pradinio elemento geometrijos dalies. Pavyzdžiui, yra 10 hektarų sklypo poligonas, kurio *Vertės* atributo reikšmė yra sklypo piniginė vertė. Dalinant šį sklypą į du naujus 3 ha ir 7 ha sklypus, pradinio sklypo *Vertės* atributo reikšmė pirmam sklypui dauginama iš 30 proc., antram – iš 70 proc.

Jungimo taisyklės nurodo, kaip iš atskirų elementų atributų gauti sujungto elemento atributo reikšmę. Galimos trys jungimo taisyklių nuostatos:

Išankstinė reikšmė: kai du ar daugiau elementų jungiami į vieną naują, jo atributams priskiriama išankstinė reikšmė, kuri gali būti nesusijusi su pradinių elementų atributų reikšmėmis. Kaip ir skaldymo taisyklėms, ši nuostata geriausia tada, kai didžiąja dauguma atvejų tinka viena reikšmė.

Reikšmių suma: jei atributas skaitinis, jungiamų elementų atributų reikšmės sudedamos, ir ši suma įrašoma į naujo sujungto elemento atributą. Jungdami du sklypus į vieną šią nuostatą pasirinktume, jei norėtume, kad naujo, didesnio sklypo *Vertė* būtų lygi sujungtų sklypų verčių sumai.

Geometrijos svoriai: jei atributas skaitinis, naujo elemento atributo reikšmė yra pradinių elementų atributų reikšmių svorinis vidurkis; reikšmės svoris lygus elemento geometrinės charakteristikos daliai naujame objekte. Tarkime, yra poligoninių elementų klasė, kurioje saugomi panašių miško savybių plotai, galbūt įvertintos ir medžių rūšys bei aukštis. Ši elementų klasė gali turėti atributą *TūrisHa*, nurodantį vienam hektarui apskaičiuotą medienos tūrį kubiniais metrais. Mes norime sujungti du poligonus: vieno plotas – 3,0 ha, *TūrisHa* – 250 m^3/ha, kito plotas – 7,0 ha, *TūrisHa* – 300 m^3/ha. Naujojo poligono *TūrisHa* bus 0,3 x 250 + 0,7 x 300 = 285 m^3/ha.

Domenai ir tikrinimo priemonės leidžia kontroliuoti įvedamas atributų vertes ir struktūruoti duomenis, o jungimo ir skaldymo taisyklės tiesiog palengvina ir pagreitina darbą su duomenimis. Jos automatiškai įveda įprastinių atributų reikšmes, ir operatoriams gali tekti keisti atributus tik nestandartiniais atvejais.

Naudojant visas elgsenos kontrolės priemones kartu, tikro pasaulio objektus elementų klasėmis galima atvaizduoti labai tiksliai. 2-5 lentelėje pateiktas pavyzdys, kaip šiomis priemonėmis galima griežtai kontroliuoti kelių elementus (adaptuota iš Zeiler, 1999, 89 p.). Čia laikoma, kad yra vienintelė kelių elementų klasė, suskirstyta į potipius pagal kelio dangos medžiagą.

2-5 lentelė. Elgsenos elementų pavyzdys

Elgsenos elementas	Pavyzdžiai
Išankstinės reikšmės	▪ Naujam betoniniam keliui suteikiamos 4 juostos ▪ Naujam asfaltuotam keliui suteikiamas 10 metrų plotis
Atributų Domenai	▪ Galimas asfaltuoto kelio plotis yra 10 m, 12 m arba 15 m, galimas juostų skaičius – 1, 2 arba 4 ▪ Galimas žvyrkelio plotis yra 5 m, 8 m arba 10 m, galimos priesagos – „Juosta“ ir „Kelias“
Skaldymo / jungimo taisyklės	▪ Sujungtas asfaltuotam keliui priskiriama išankstinė reikšmė – 2 juostos ▪ Suskaldytas asfaltuotas kelias lieka to paties pločio
Jungimosi taisyklės	▪ 2 juostų asfaltuotas kelias gali jungtis tik su kitu 2 juostų keliu ▪ Betoninis kelias gali jungtis su asfaltuotu keliu, bet ne su žvyrkeliu
Ryšių taisyklės:	▪ Asfaltuotą kelią galima susieti su tunelių arba tiltų elementais ▪ Žvyrkelių sankirtos taške negali būti daugiau negu 4 kelių

Topologija

Topologija yra priemonė erdviniams ryšiams tarp elementų ar elementų klasių geoduomenų bazėje kontroliuoti. Pagrindinė domenų paskirtis – užtikrinti atributinių duomenų vientisumą, o topologijos taisyklės padeda užtikrinti erdvinių duomenų vientisumą (kokybę). Geoduomenų bazėse topologija yra trijų tipų.

Žemėlapio topologija: *ArcGIS* programos vykdoma topologijos kontrolė redagavimo seanso metu. Kaip išsiaiškinome nagrinėdami SHAPE failus, žemėlapio topologija gali užtikrinti, kad SHAPE failai atitiks topologijos taisykles, pavyzdžiui, linijų galai bus sujungti, o poligonai turės bendras ribas. Šio tipo topologija niekada nesaugoma kartu su galutiniais duomenimis – tai tiesiog redagavimo įrankių rinkinys, leidžiantis naudotojams geriau struktūruoti duomenis.

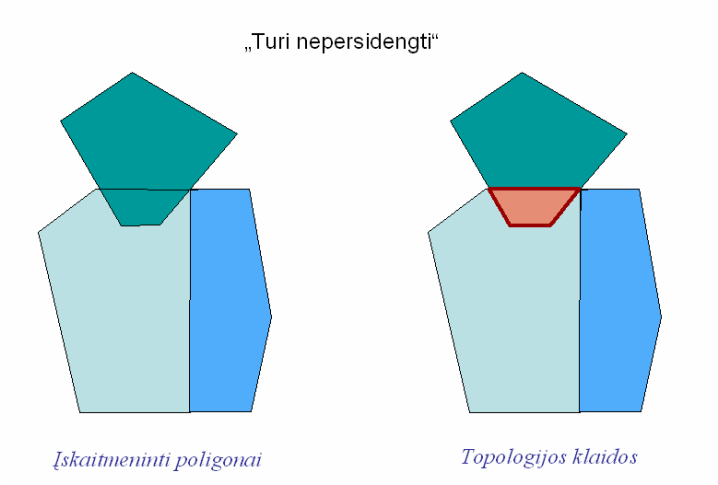
Geoduomenų bazės topologija: rinkinys taisyklių, apibrėžiančių erdvinius ryšius tarp vienos ar kelių elementų. klasių Tai yra geoduomenų bazėje saugomas elementas, kurį galima pridėti prie rodinio kaip ir kitas elementų klases. Geoduomenų bazės topologija leidžia *ArcGIS* programai tikrinti pridedamus arba redaguojamus elementus ir užtikrinti, kad jie atitiktų erdvines taisykles.

Geometrinio tinklo topologija:	skirta geometrinį tinklą sudarantiems objektams struktūruoti. Ji leidžia *ArcMap* programai suprasti tinklo jungimąsi ir srautus. Geometriniai tinklai turi briaunų ir jungčių elementus; jie naudojami upių, vandentiekio, nuotekų, elektros ir kitų tinklų srautams modeliuoti.

Geoduomenų bazės topologijos taisykles galima sudaryti vienos klasės arba skirtingų klasių elementams. Pavyzdžiui, galima sukurti taisyklę, kad žemės sklypai negali persidengti, arba kad visas pastatas turi būti vieno sklypo viduje.

Su geoduomenų bazių topologija paprastai dirbama taip: pirmiausia apibrėžiamos elementų klasių erdvinę elgseną kontroliuojančios taisyklės, po to tikrinama elementų geometrija. Pavyzdžiui, galima nustatyti, kad žemės sklypai negali persidengti. Apibrėžus šią taisyklę, tikrinami sklypai, ir visos persidengiančios sritys pateikiamos naudotojui taisyti. Taigi elementų topologija (pavyzdžiui, gretimi arba persidengiantys sklypai) tiesiogiai duomenų bazėje nesaugoma, kaip coverage sluoksnių atveju, bet nustatoma arba kai kuriamos taisyklės, arba kai prie elementų klasių pridedami nauji elementai.

Programoje yra 26 išankstinės topologijos taisyklės, kiekviena jų nurodo skirtingą tinkamo topologinio ryšio tipą (visos taisyklės aiškiai aprašytos *ArcGIS* žinyne, http://webhelp.esri.com/arcgisdesktop/9.2/index.cfm?TopicName=Topology_rules).
Pavyzdžiui, taisyklė „turi nepersidengti“ (*Must Not Overlap*) taikoma tos pačios poligoninės klasės elementams ir tikrina, kad nebūtų erdvėje persidengiančių poligonų. Poligonai gali turėti bendras ribas ar liestis viename taške, bet persidengti jie negali. 2-26 pav. pavaizduoti trys elementai, įskaitmeninti taip, kas du iš jų turi bendrą vertikalią ribą, o trečias persidengia su vienu iš pirmų dviejų. Tikrinant šiuos elementus pagal taisyklę „turi nepersidengti“ (*Must Not Overlap*), persidengiančios dalys pripažįstamos topologijos klaidomis ir paryškinamos ekrane. Atkreipkite dėmesį, kad nelaikoma klaida, jei poligonai turi bendras kraštines ar liečiasi taške.

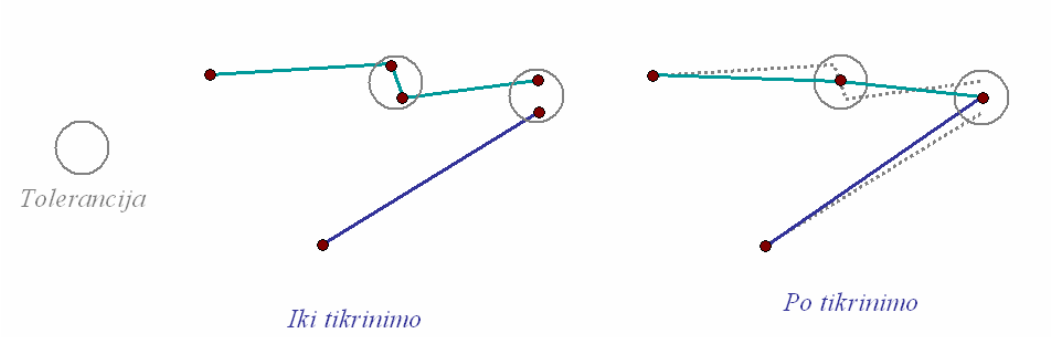


2-26 pav. Topologijos klaidos tikrinant pagal taisyklę „turi nepersidengti“

Apibrėžiant topologijos taisyklę, nurodoma klasterio tolerancija (**Cluster Tolerance**). Klasterio tolerancija – tai atstumas, kuriuo esantys visi taškai laikomi vienu tašku. Šiuo atstumu esantys taškai sutraukiami į vieną tašką. Tai galioja ir greta esančioms to paties

linijinio ar poligoninio elemento viršūnėms, ir dviejų greta esančių elementų viršūnėms. Klasterio tolerancijos paskirtis – sutapdinti taškus, kurie turėtų sutapti, bet yra galbūt tik labai arti vienas kito, prieš nustatant gretimus, persidengiančius ir susikertančius elementus. Klasteriai neskirti elementams generalizuoti.

2-27 pav. pavaizduoti du linijiniai elementai. Klasterio tolerancijos atstumas pavaizduotas pilku apskritimu. Atkreipkite dėmesį, kad iki tikrinimo dviejose vietose į apskritimą patenka daugiau negu vienas taškas. Atliekant tikrinimą, visi tolerancijos apskritime esantys taškai sutraukiami į vieną tašką, esantį vidutiniame pradinių taškų centre. Šiuo atveju tikrinant bus pakeista šviesiai mėlynos linijos forma, ir abi linijos bus sujungtos vienoje viršūnėje.

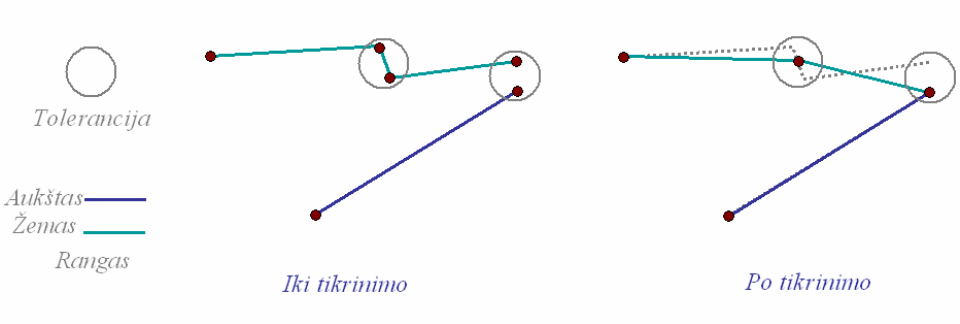


2-27 pav. Klasterio tolerancijos įtaka tikrinant topologiją

Klasterio tolerancijos turi būti labai mažos. Išankstinė tolerancijos reikšmė yra 0,001 metro, taigi bus perkelti tik labai arti vienas kito esantys taškai. Nurodant klasterio toleranciją, reikia rasti kompromisą: nurodysite per mažą – elementai liks netinkamai integruoti, nurodysite per didelę – gali nepageidaujamai pakisti elementų forma. Bendru atveju klasterio tolerancija turi būti daug mažesnė už erdvinį duomenų tikslumą. Pavyzdžiui, jei elementų tikslumas yra 2,0 metro, didžiausia dėmesio verta tolerancija turėtų būti visa eile mažesnė (0,2 metro). Dažnai tolerancija pasirenkama keliomis eilėmis mažesnė už duomenų tikslumą (galbūt 0,02 arba 0,002 metro).

Apibrėždamas topologijos taisyklę, naudotojas gali nurodyti elementų klasių rangus. Elementų klasių **rangas** santykinai apibūdina elementų klasių padėties svarbą. Jei atstumas tarp dviejų elementų mažesnis už klasterio tolerancijos atstumą, rangai nurodo *ArcGIS* programai, kurį elementą galima judinti, o kurio ne. Rangą galima priskirti visoms topologiniais ryšiais susietoms elementų klasėms; jei atstumas tarp dviejų viršūnių bus mažesnis už klasterio toleranciją, žemesnio rango elementas bus pritrauktas prie aukštesnio rango elemento.

Įsivaizduokime, kad ankstesniame pavyzdyje nagrinėti elementai turi skirtingus rangus. Ši situacija pavaizduota 2-28 pav. Šviesiai mėlynos linijos forma pakis kaip ir anksčiau; bet linijų galai bus sujungti kitaip, nes elementų klasės turi rangus. Tamsiai mėlynos linijos rangas aukštesnis, todėl ji liks savo vietoje, o šviesiai mėlyna linija (žemesnio rango elementas) bus perkelta ir pritraukta prie tamsiai mėlynos. Taigi aukštesnio rango elemento (tamsiai mėlynos linijos) geometrija visai nepasikeis.

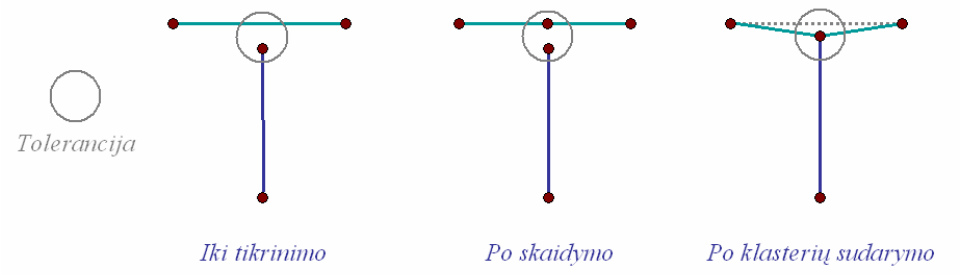


2-28 pav. Elementų klasių rangų įtaka tikrinant topologiją

Elementų klasių rangus patartina naudoti, tikrinant pagal topologijos taisykles skirtingo tikslumo duomenis. Bendru atveju mažesnio tikslumo elementų klasės rangas turi būti žemesnis negu didelio tikslumo elementų klasės, kad didžioji tiksliausių duomenų dalis liktų nejudinta.

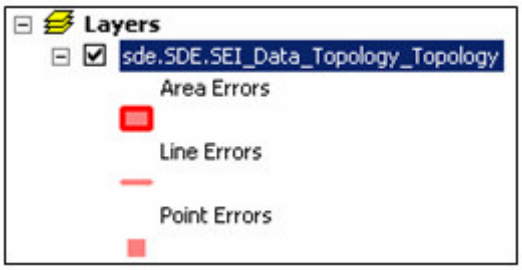
Topologijos tikrinimo procesas panašus į dviejų etapų elementų geometrijos keitimo procesą. Pirmasis etapas yra skaidymas (**Cracking**): į linijinį elementą įterpiamos naujos viršūnės tose vietose, šalia kurių mažesniu negu tolerancija atstumu yra kitų taškų, bet anksčiau viršūnių nebuvo. Antras etapas – klasterių sudarymas (**Clustering**): visi taškai, atstumas tarp kurių ne didesnis už klasterio toleranciją, sutraukiami į vieną tašką.

2-29 pav. pavaizduotos dvi linijos, atstumas tarp kurių *iki tikrinimo* ne didesnis už klasterio toleranciją. Tačiau viršuje esanti šviesiai mėlyna linija neturi viršūnės šalia tamsiai mėlynos linijos. Skaidymo procesas įterpia į šviesiai mėlyną liniją naują viršūnę arčiausiai tamsiai mėlynos linijos esančiame taške. Tada klasterių sudarymo procesas sutraukia dvi viršūnes į vieną – pritraukia abu linijinius elementus prie šių viršūnių vidutinio centro.



2-29 pav. Skaidymas ir klasterių sudarymas tikrinant topologiją

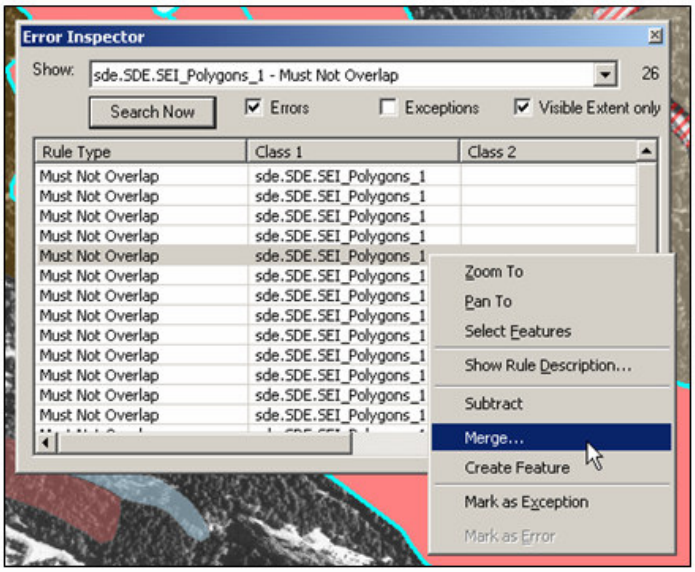
Po tikrinimo topologiją galima pridėti prie *ArcMap* rodinio, ir naudotojas gali po vieną pataisyti visas nustatytas klaidas. 2-30 pav. pavaizduota topologija, pridėta prie *ArcMap* turinio. Atkreipkite dėmesį, kad kiekvienas klaidos tipas (poligonų, linijų, taškų) vaizduojamas skirtingu simboliu.



2-30 pav. Topologija *ArcMap* turinio lange

Sukurta įvairių įrankių, padedančių naudotojui susitvarkyti su topologijos klaidomis:

Klaidų naršyklė: (*Error Inspector)* – dialogas, kuriame išvardijamos visos aptiktos klaidos; galima surasti kiekvieną klaidą ir padidinti jos vaizdą ekrane. Naudotojui pateikiami veiksmai, leidžiantys greitai ištaisyti topologines klaidas. 2-31 pav. pavaizduota klaidų naršyklės sąsaja.

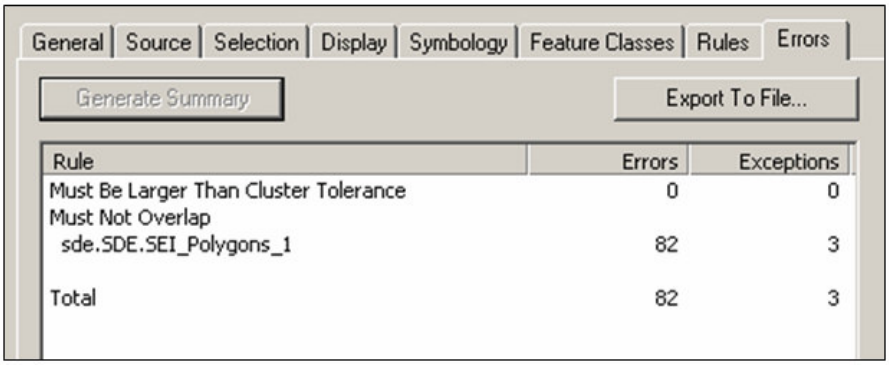


2-31 Topologijos klaidų naršyklė

Šiame pavyzdyje pažymėta viena klaida iš viso klaidų sąrašo. Spustelėjus ją dešiniuoju pelės mygtuku, naršyklė pateikia tris sprendimus, kaip galima sutvarkyti poligonų perdangą: atimti (*Subtract*) – perdangos sritis bus paprasčiausiai pašalinta; sujungti (*Merge*) – perdangos sritis bus pridėta prie vieno iš persidengiančių poligonų ir atimta iš visų kitų, kuriems ji taip pat priklauso; sukurti elementą (*Create Feature*) – iš perdangos srities bus paprasčiausiai sukurtas naujas poligonas, ir ši sritis pašalinta iš visų persidengiančių poligonų. Siūlomos pataisos priklausys nuo topologinės klaidos pobūdžio. Jei siūlomi variantai netinka, naudotojai gali pataisyti elementus rankiniu būdu.

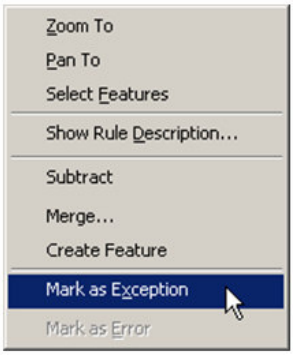
Topologijos įrankių juosta: leidžia greitai pasiekti kelis su topologija susijusius įrankius, įskaitant topologijos tikrinimo mygtukus ir klaidų naršyklės mygtuką. Ši įrankių juosta yra *ArcMap* sąsajos dalis.

Klaidų ataskaitos: parodo ekrane arba įrašo į failą klaidų ataskaitą, kurioje susumuojamos visos nesutvarkytos topologijos klaidos pagal kiekvieną topologijos taisyklę. Šią ataskaitą naudinga pridėti prie duomenų dokumentacijos užbaigus projektą: ji rodys, kad visos topologijos klaidos buvo sėkmingai pašalintos. 2-32 pav. pateiktas klaidų ataskaitos pavyzdys.



2-32 pav. Topologijos klaidų ataskaitos pavyzdys

Nors topologijos klaidos dažnai atsiranda dėl duomenų klaidų, kartais erdvinės taisyklės iš tikro gali būti pažeidžiamos. Tokiu atveju taisykles pažeidžiantį elementą reikia pažymėti kaip išimtį (**Exception**) – tai reiškia, kad šiuo konkrečiu atveju mes sąmoningai paliekame topologinę klaidą. Pavyzdžiui, gali būti topologijos taisyklė, reikalaujanti, kad bet kuris pastatas turi būti vieno sklypo viduje, ir negali kirsti sklypo ribos bei pažeisti kaimynų nuosavybės teisių. Nors ši taisyklė teisinga didžiąja dauguma atvejų, kartais gali pasitaikyti pastatų, per klaidą pastatytų ant sklypo ribos, arba tam pačiam savininkui gali priklausyti du gretimi žemės sklypai, ir pastatą jis tyčia pastatė abiejuose sklypuose. Tokiais retais atvejais geriausia pažymėti, kad šie elementai yra išimtys. Taisydamas klaidas, operatorius gali tiesiog spustelėti klaidą dešiniuoju mygtuku ir pažymėti *Mark as Exception* (žymėti kaip išimtį), kaip pavaizduota 2-33 pav.



2-33 pav. Išimties žymėjimas

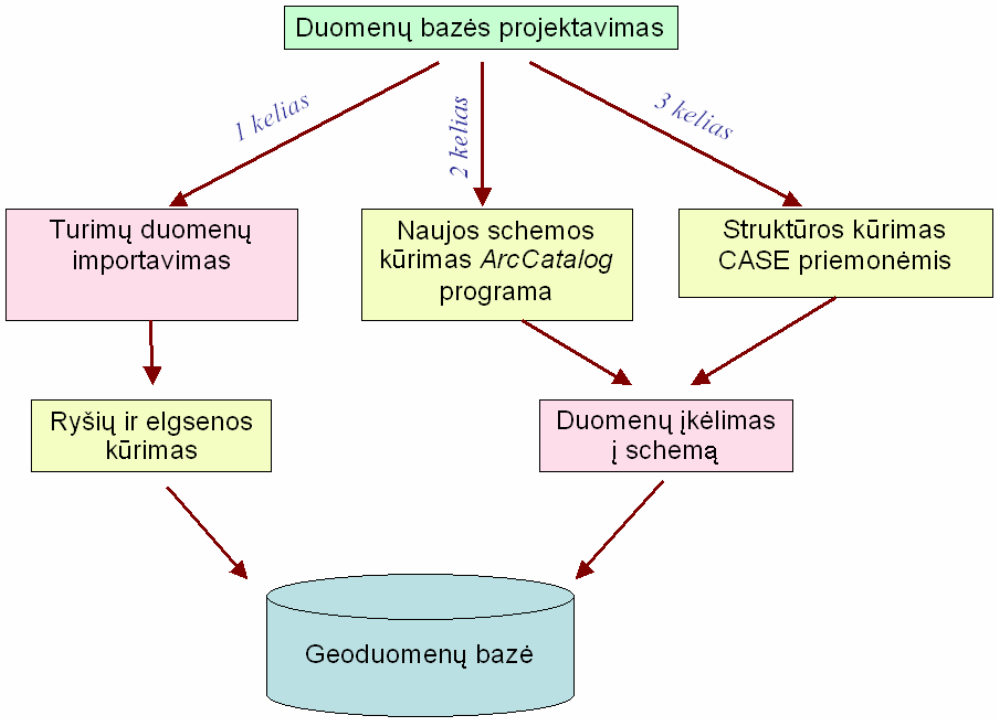
2.4 Geoduomenų bazės kūrimas

2.4.1 Kūrimo procesas

Naują geoduomenų bazę sukurti galima keliais būdais. Kūrimo procesas visada turėtų prasidėti projektavimo etapu, bet nuo projekto iki užbaigtos geoduomenų bazės su elementų klasėmis ir domenais gali vykti įvairiai.

Iš esmės yra du geoduomenų bazės kūrimo būdai. Galima importuoti turimus duomenis tokius, kokie yra, o ryšius, domenus ir kitus elementus pridėti vėliau; arba galima sukurti tuščią duomenų bazę su visais reikalingais ryšiais bei elgsenomis ir tada importuoti arba sukurti reikiamus duomenis.

Kūrimo procesą galima apibrėžti taip, kaip pavaizduota 2-34 pav. Matyti, kad procesas visada prasideda projektavimo etapu ir baigiasi užbaigta geoduomenų baze. Bet nuo proceso pradžios iki pabaigos galimi įvairūs keliai.



2-34 pav. Geoduomenų bazės kūrimo procesas

Geltoni langeliai yra užduotys, susijusios su geoduomenų bazės struktūros arba schemos kūrimu ar keitimu. Šiais etapais apibrėžiami duomenų bazės elementai – atributai, ryšiai, potipiai, domenai ir topologija. Rožiniai langeliai – su duomenų įkėlimu susijusios užduotys. Matyti, kad kiekviename kelyje yra arba įkėlimo procesas, po kurio seka schemos tobulinimas, arba schemos apibrėžimo procesas, po kurio seka duomenų įkėlimo fazė.

Tris pagrindiniai keliai nuo duomenų bazės projekto iki užbaigtos geoduomenų bazės pavaizduoti 2-34 pav.

1 kelias: importuoti turimus duomenis į naują geoduomenų bazę, tada geoduomenų bazę patobulinti – *ArcCatalog* programa pridėti ryšius ir elgsenos elementus.

2 kelias: *ArcCatalog* programa rankiniu būdu sukurti suprojektuotą duomenų bazę – elementų klases, atributus, ryšius ir elgseną, tada įkelti duomenis į tuščią schemą arba juos sukurti.

3 kelias: Kompiuterizuoto informacijos sistemų projektavimo priemonėmis (*Computer Aided Software Engineering* – CASE) suprojektuoti geoduomenų bazės struktūrą, tada tiesiai iš šio projekto ESRI priemonėmis automatiškai sukurti duomenų bazę. Šiuo atveju CASE programinė įranga intensyviai naudojama ir pačiu projektavimo etapu, bet kad būtų aiškiau, projektavimo dalį atskyrėme nuo praktinio įgyvendinimo dalies. Kaip ir 2 keliu, šiuo keliu kuriama tuščia geoduomenų bazės struktūra, į kurią vėliau įkeliami duomenys.

Praktiškai organizacijos gali naudoti kai kuriuos arba visus šiuos metodus.

Dauguma organizacijų turi didelius įvairių formatų duomenų kiekius, kuriuos nori įkelti į duomenų bazę. Tai gali būti coverage sluoksniai, SHAPE failai, kompiuterizuoto projektavimo (CAD) failai ir *dBase* arba *Excel* formato neerdvinių duomenų lentelės. Turimų duomenų importavimas visada sudaro didelę geoduomenų bazės kūrimo proceso dalį.

ArcCatalog ir *ArcToolbox* turi priemonių šiems duomenims importuoti, ir patogius vedlius įvairioms užduotims atlikti, pavyzdžiui, įvesties lentelės atributams sutapdinti su galbūt kitaip pavadintais paskirties lentelės atributais. Be to, šios priemonės užtikrina, kad vieno formato duomenų tipai ar geometrijos tipai bus sutapdinti su tinkamais geoduomenų bazės elementų tipais. Tarkime, kad reikia pakeisti įvedamo SHAPE failo *dBase* skaičių tipo (*Number*) duomenų lauką į trumpo arba ilgo sveikojo skaičiaus geoduomenų bazės lauką.

Įkeliant į geoduomenų bazę lenteles arba elementų klases, patartina naudotis ne RDBVS priemonėmis, o su ESRI programine įranga pateiktomis priemonėmis. Naudojant *ArcMap* įrankius, nauji objektai bus užregistruoti geoduomenų bazėje, ir juos bus galima susieti ryšiais, tikrinti pagal taisykles ir pan.

2.4.2 CASE priemonės

Kompiuterizuotas informacijos sistemų projektavimas (CASE) – tai programų ir duomenų bazių projektavimas programinėmis priemonėmis. Šios programinės priemonės dažnai vadinamos CASE priemonėmis. CASE programos paprastai gali modeliuoti duomenis ir duomenų bazių bei sistemų elgseną; dažnai tai atliekama UML kalba. Kuriant taikomąsias programas, CASE priemonėmis dažnai galima sugeneruoti dalį kodo. Kuriant duomenų bazių programas, CASE priemonėmis galima sukurti duomenų bazių struktūrą ir sugeneruoti kodą, padedantį tvarkyti elgsenas ar apribojimus. Paprastai jos taip pat turi programavimo darbų valdymo funkcijų, įskaitant versijų kontrolę. Bendrai CASE įrankiais galima laikyti visas programines priemones, naudojamas įvairiais sistemų kūrimo ir plėtros ciklo etapais. Tai projektų valdymo, funkcinės analizės, sistemų projektavimo, vertimo, bandymų, naudotojo dokumentacijos generavimo ir panašios programos. Mes išsamiai

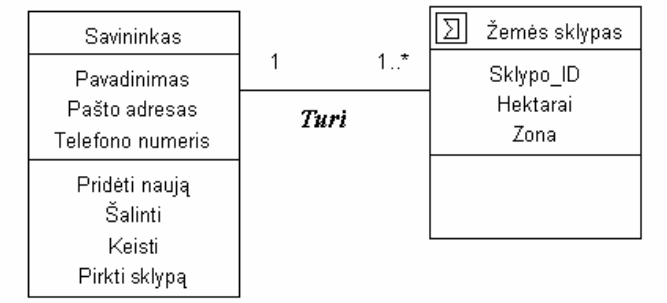
nagrinėsime priemones, padedančias projektuoti duomenų bazes UML klasių diagramomis, taip pat šių projektų konvertavimo į veikiančias duomenų bazes procesą.

Geoduomenų bazių projektavimas CASE priemonėmis turi kelis pranašumus:

- projektas yra ir geoduomenų bazės dokumentacija; jį patogu kurti keliese ar nusiųsti peržiūrėti kitiems;
- didelių arba sudėtingų geoduomenų bazių modelius sukurti galima greičiau arba aiškesnius;
- lengva atnaujinti duomenų bazės struktūrą – projekto pokyčius galima greitai įdiegti į geoduomenų bazę, net jei duomenys jau įkelti.

Kuriant geoduomenų bazę CASE įrankiais, naudojami trys komponentai: projekto ženklų sistema, kuria aprašomas duomenų bazės modelis, projektavimo programinė įranga, kuria projektas kuriamas, ir mechanizmas projektui konvertuoti į užbaigtą, paruoštą naudoti duomenų bazę. Šiuo atveju ženklų sistema bus UML klasių diagramos, kurias išsamiai aptarėme 1 kurso dalyje. Yra keli projektavimo programų paketai, o ESRI palaiko du pagrindinius (*Rational Rose* ir *Visio*). Mes pavyzdžiams ir aptarimui naudosime *Microsoft Visio*. *Visio* yra bendros paskirties braižymo programa, kuria galima kurti blokines schemas, organizacines schemas ir net biuro išdėstymo planus. Ja taip pat galima kurti įvairias UML diagramas – pavyzdžiui, užduočių (*Use Case Diagrams*), sekų (*Sequence Diagrams*) ir klasių (*Class Diagrams*). Importo mechanizmas yra *ArcCatalog* schemos vedlys (*Schema Wizard*).

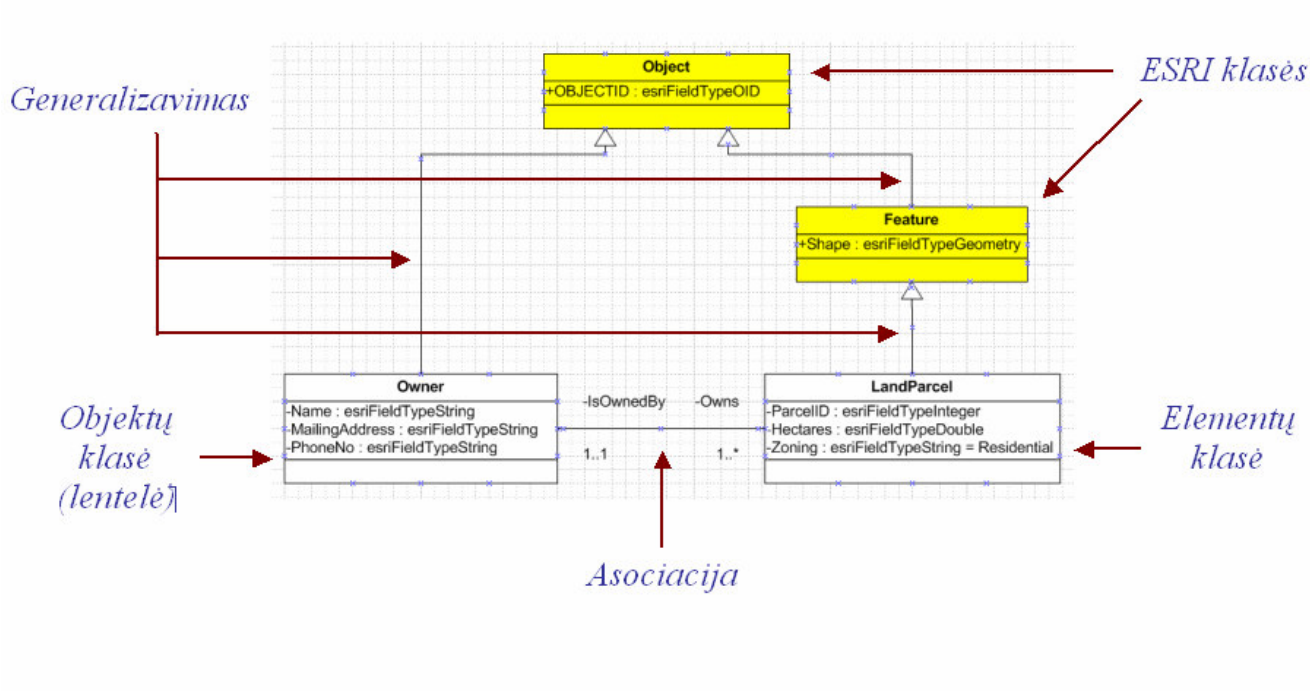
Geoduomenų bazės kūrimo *Visio* programa pavyzdį paimsime iš 1 kurso dalies – tai pavyzdys apie savininkų-žemės sklypų ryšį. 2-35 pav. dar kartą pateikiame šio ryšio klasių diagramą. Tai paprastas ryšys tarp neerdvinės *Savininkų* lentelės ir poligoninių *ŽemėsSklypų* elementų klasės. Šios dvi klasės atributų turi nedaug, kad būtų paprasčiau parodyti, kaip jie bus praktiškai įgyvendinti *Visio* programoje.



2-35 pav. Savininkų-sklypų ryšys

2-36 pav. pavaizduota, kaip ta pati klasių diagrama atrodytų nubraižyta *Visio* programoje. Iškart matyti aiškus skirtumas: atsirado dvi papildomos geltonos klasės – objektų (*Object*) ir elementų (*Feature*). Jos vaizduoja ESRI *ArcObjects* klases, kurių pagrindu bus įgyvendintos dvi mūsų pačių sukurtos klasės. *ŽemėsSklypų* klasė susieta su *Elementų* klase paveldėjimo arba apibendrinimo ryšiu. Jei prisimenate, šis ryšys dažnai vadinamas asociacija „yra“ (is-a), taigi *ŽemėsSklypas* yra *Elementas*. Tai leidžia *ArcGIS* programai suprasti, kad naujos geoduomenų bazės *ŽemėsSklypų* klasė turi būti poligoninių elementų klasė. Jei šių ESRI objektų apibrėžimų nebūtų, *ArcGIS* negalėtų konvertuoti *Visio* klasių į

naujos geoduomenų bazės elementus. Be to, šioje diagramoje *Savininkų* lentelė susieta su *Objektų* klase, kad *ArcGIS* suprastų, jog *Savininkų* lentelė yra neerdvinė; *Elementai* sujungti su *Objektais* – tai reiškia, kad elementų klasė yra tiesiog patobulinta objektų klasė, turinti erdvinį komponentą.

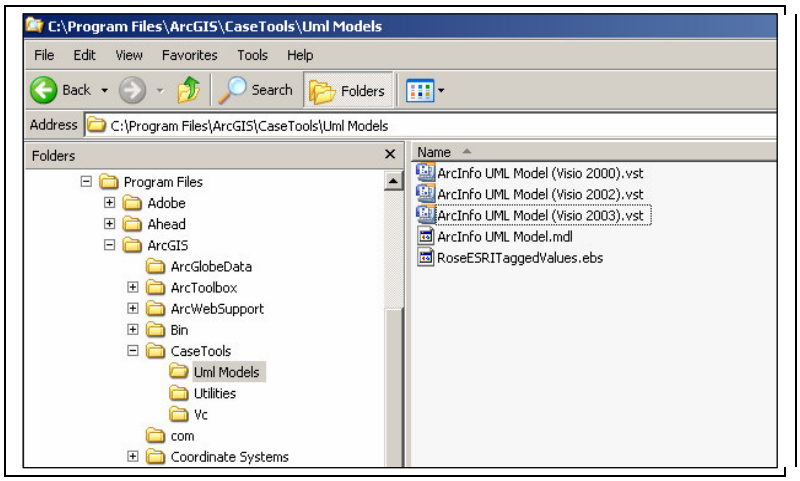


2-36 pav. *Visio* projekto pavyzdys

Taip pat atkreipkite dėmesį, kad atributai apibrėžti detaliau. Kiekvienam atributui suteiktas duomenų tipas, kuris taip pat turi atitikti vieną iš ESRI duomenų tipų. Štai kodėl atributai yra esriFieldTypeDouble ir esriFieldTypeString tipo. Kaip ir *Elementų* bei *Objektų* klasės, šie ESRI duomenų tipai leidžia sukurti teisingus duomenis palaikančią geoduomenų bazę. Be to, *ŽemėsSklypo Zonos* atributui suteikta išankstinė reikšmė – „Gyvenamoji“ (*residential*). *Visio* programoje galima apibrėžti daugumą anksčiau išnagrinėtų geoduomenų bazių dalių. Galima sukurti potipius, išankstines reikšmes ir domenus. Kai kurių dalių, pavyzdžiui, topologijos, erdvinių charakteristikų ir anotacijų *Visio* programa sumodeliuoti neįmanoma – prie duomenų modelio juos reikia pridėti *ArcCatalog* programa po to, kai duomenų bazė bus sukurta.

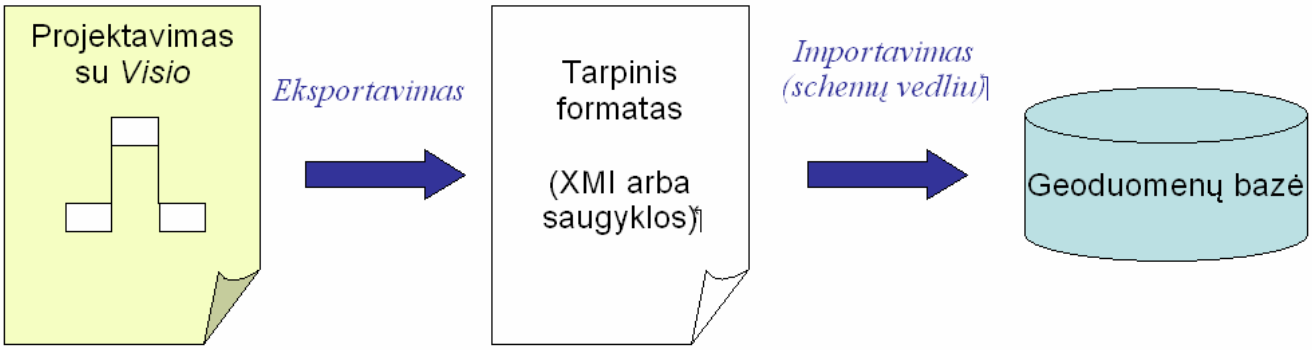
Elementų klasė, laukų duomenų tipai ir kiti ESRI objektai apibrėžti *Visio* šablono faile, įdiegtame į jūsų kompiuterį kartu su *ArcGIS*. Prieš pradedant projektą, reikia įkelti šį šabloną, kad ESRI objektai būtų įtraukti į *Visio*. Šablono faile be kitų *ArcObjects* elementų yra *Objektų* ir *Elementų* klasės, geoduomenų bazės laukų duomenų tipai, geometrijos tipai ir specialios klasės, pavyzdžiui, tinklo briaunų ir jungčių elementų klasės. Jie nurodo, kaip *Visio* projektą paversti užbaigta geoduomenų baze.

Skirtingoms *Visio* ir *Rational Rose* versijoms sukurti atskiri šablonai. Šie failai yra *ArcGIS* įdiegimo aplanke. 2-37 pav. pavaizduoti su *ArcGIS* 9.2 pateikti šablonų failai ir jų vieta tipinio įdiegimo atveju.



2-37 pav. ESRI šablonų failai

Jau minėjome, kad iš užbaigto projekto reikiamas geoduomenų bazės dalis sukuriantis importo mechanizmas yra *ArcCatalog* programos schemos vedlys (*Schema Wizard*). Schemos vedlys negali tiesiogiai skaityti *Visio* projektų – juos reikia konvertuoti į tarpinį formatą. Tai gali būti XML duomenų mainų (XML *metadata interchange* – XMI) arba *Microsoft Repository MDB* formatas. Taigi, perėjimas nuo projekto prie geoduomenų bazės vyks pagal 2-38 pav. pateiktą schemą.



2-38 pav. CASE priemonių procesas

Schemos vedlys perskaitys XMI arba saugyklos duomenų bazę ir nustatys, kokius objektus, ryšius ir elgsenos elementus reikia sukurti naujojoje geoduomenų bazėje. Importo proceso metu naudotojai gali nustatyti tam tikras savybes, pavyzdžiui, tam tikrų elementų klasių arba elementų duomenų rinkinių erdvines charakteristikas. Importavę savo mažą projektą, gausime paprastą duomenų bazę, pavaizduotą 2-39 pav. Užbaigta geoduomenų bazė turi vieną poligoninių *ŽemėsSklypų*elementų klasę, neerdvinę *Savininkų* lentelę ir jas siejančią ryšių klasę „*Turi*“. Visi šių elementų atributai ir kartotinumai sukuriami importuojant projektą.

2-39 pav. Užbaigta Geoduomenų bazė

Daugelio GIS taikymo sričių, pavyzdžiui, miškininkystės, geologijos, požeminio vandens ir transporto duomenų modeliai jau yra sukurti. ESRI prižiūri kelis šiems modeliams skirtus tinklalapius, skatina naudotis šiais modeliais ir remia jų tobulinimą. Šie duomenų modeliai gali būti išeities tašku tolesniam darbui jūsų pasirinktoje srityje arba iliustracija, kaip skirtingais būdais galima atvaizduoti įvairius objektus geoduomenų bazėje.

Literatūra

- Blaha, M., and Premerlani, W. *Object-Oriented Modeling and Design for Database Applications*. Prentice-Hall, 1998.
- ESRI. *Building a Geodatabase*. ESRI Press, 2005.
- ESRI. *Introduction to CASE Tools*. ESRI Press, 2004.
- ESRI. *Working with Geodatabase Topology*. ESRI Press, 2003.
- Kroenke, D.M. *Database Processing Fundamentals, Design and Implementation*. Prentice-Hall, 1998.
- Shekhar, S., and Chawla, S. *Spatial Databases: A Tour*. Prentice-Hall, 2003.
- Zeiler, M. *Modeling Our World*. Redlands, CA: ESRI Press, 1999.

3 Keliems vartotojams skirtos geoduomenų bazės

Keliems vartotojams tuo pačiu metu suteikiant galimybę naudoti duomenų bazę gali atsirasti naujų duomenų bazės valdymo problemų. Šioje dalyje pateiksime bendrą proceso suteikiant keliems vartotojams prieigą prie duomenų apžvalgą bei metodą, kurį naudodamas ESRI taikomųjų programų paketas sprendžia kilusius sunkumus.

Pirmoje šios dalies temoje pateikiama teorinių keliems vartotojams skirtos duomenų bazės pagrindų apžvalga, įskaitant galimų problemų aprašą ir galimus šių problemų sprendimo būdus. 2 temoje aptariama, kaip ESRI taikomųjų programų paketas tvarko kelių vartotojų prieigą. Joje apibūdinama *ArcSDE* (*Spatial Database Engine*) programa ir mechanizmas, kurį ji naudoja palaikyti kelis tuo pačiu metu vykstančius redagavimus (versijas). Paskutinėje temoje nagrinėjamas geoduomenų bazių tvarkymas. Nors paskutinė tema nėra skirta tik kelių vartotojų prieigai, daug geoduomenų bazės tvarkymo klausimų kyla dėl kelių vartotojų palaikymo ir būdingų problemų, atsirandančių dėl geografinių duomenų pakeitimų.

Dalies turinys

- Keliems vartotojams skirtos duomenų bazės
- *ArcSDE* ir versijos
- Geoduomenų bazės tvarkymas

3.1 Keliems vartotojams skirtos duomenų bazės

Kelių vartotojų duomenų bazės apdorojimas vyksta tada, kai duomenų bazę vienu metu naudoja daugiau negu vienas asmuo. Beveik visos didelės organizacijos, naudojančios GIS technologiją, norės, kad keli vartotojai galėtų pasiekti tuos pačius duomenis tuo pačiu metu. Ši situacija gali labai skirtis nuo vienam vartotojui skirtų duomenų bazių sistemų naudojimo. Viena didžiausių problemų yra galimybė, kad vieno asmens veiksmai prieštaraus kito asmens veiksmams. Pavyzdžiui, du pardavėjai gali tuo pačiu metu parduoti tą pačią prekę, jeigu nėra nustatoma, kaip tvarkoma prieiga prie to paties lentelėje esančio įrašo. Be to, prie duomenų bazės gali jungtis daug skirtingos informacijos reikalaujančių ir įvairius darbo įgūdžius turinčių vartotojų, todėl gali prireikti tvarkyti duomenų bazę, pavyzdžiui, jos saugą. Didesnėse organizacijose gali prireikti papildomų darbuotojų, kurie atliktų duomenų bazės tvarkymo (DBT) užduotis.

Terminas **vienalaikiškumas** (*concurrency*) apibūdina metodus, kuriuos naudojant keli vartotojai tuo pačiu metu gali gauti prieigą prie tos pačios duomenų bazės. DBVS procedūros, užtikrinančios, kad tuo pačiu metu vykdomos duomenų bazės transakcijos neprieštaraus viena kitai, kartais vadinamos vienalaikiškumo valdymo mechanizmais. Tokie mechanizmai bando pateikti vieno vartotojo darbo rezultatus keliems vartotojams skirtoje aplinkoje taip, lyg tas vartotojas būtų vienintelis duomenų bazės vartotojas. Keliems vartotojams skirtos duomenų bazės patikimumas priklauso nuo vienalaikiškumo valdymo mechanizmų ir duomenų bazės **atkūrimo** funkcijų, kai galima pataisyti arba pertvarkyti duomenis, jeigu kiltų sistemos nesklandumų.

Vienalaikiškumas ir atkūrimas yra plačios temos. Šios temos tikslas – supažindinti studentus su terminija ir svarbiausiomis sąvokomis.

3.1.1 Transakcijų apdorojimas

Naudodami duomenų bazes vartotojai dažniausiai atlieka **transakcijas** (*transactions*), kurios yra loginiai darbo, kuris turi būti užbaigtas arba nutrauktas, vienetai. Tarpinis rezultatas nepriimtinas. Transakcijas gali sudaryti viena arba daugiau duomenų bazės prieigos operacijų, pvz., įterpimai, panaikinimai, modifikavimai ar nuskaitymai. Įsivaizduokite situaciją, kai bankininkystės duomenų bazės vartotojas nori pervesti lėšas iš vienos banko sąskaitos į kitą. Tokiu atveju transakcija susideda iš dviejų atnaujinimų: vienu atnaujinimu sumažinamas pirmos sąskaitos likutis, o antru padidinamas paskirties sąskaitos likutis. Būtų nepriimtina, jeigu būtų atliktas tik vienas iš šių dviejų atnaujinimų, todėl visa transakcija turi būti baigta arba, kilus problemoms, nutraukta. Nebaigtos transakcijos gali kelti pavojų duomenų bazės vientisumui.

Transakcijos yra svarbios, jeigu duomenų bazę naudoja keli vartotojai, bet jos yra reikšmingos ir tad, kai duomenų bazę naudoja vienas vartotojas. Deja, visada yra galimybė, kad apdorojant transakciją įvyks DBVS klaidų. Klaidų gali kilti dėl informacijos saugojimo laikmenų gedimų ir elektros tiekimo sutrikimų. Sistemos gedimas tarp dviejų atnaujinimų anksčiau minėtos bankininkystės transakcijos metu sukeltų nepastovią duomenų bazės būseną. DBVS transakcijų valdymas užtikrina, kad nepavykus sėkmingai baigti transakcijos, bet kuri transakcijos operacija galėtų būti anuliuota.

Apdorojant transakcijas svarbiausios operacijos yra **patvirtinti** (*commit*) ir **atšaukti** (*rollback*). Patvirtinimo operacija nurodo sėkmingą transakcijos užbaigimą. Patvirtinus kelias duomenų bazės operacijas duomenų bazė bus pastovios būsenos ir visi atnaujinimai gali būti priskirti pastoviems. Patvirtinimas nurodo transakcijos pabaigą. Atšaukimo operacija anuliuos visus pakeitimus, atliktus po vėliausio patvirtinimo. Sistemos klaidos paleis atšaukimo operaciją arba atšaukimus galima paleisti naudojant programinį kodą, kaip atkūrimo procedūros dalį. Patvirtinimo ir atkūrimo operacijos labai svarbios reliacinėse sistemose, nes duomenų bazėje saugoma informacija skaidoma į daug susijusių lentelių, todėl pakeitus vieną elementą gali tekti atnaujinti kelias lenteles.

Dar kartą įsivaizduokite du atnaujinimus, kurie būtini norint pervesti lėšas iš vienos banko sąskaitos į kitą. Jei vieno atnaujinimo metu bus nuskaičiuota suma iš pirmosios sąskaitos ir įvyks kokia nors klaida, pirmasis atnaujinimas bus anuliuotas ir abi sąskaitos atkurtos į prieš pervedimą buvusią būseną. Jeigu pirmojo atnaujinimo metu bus nuskaičiuotos lėšos, o antrojo atnaujinimo metu lėšos bus pridėtos prie paskirties sąskaitos ir abu atnaujinimai bus sėkmingi, bus patvirtinti abu atnaujinimai ir priskirti pastoviesiems. Taip užtikrinsime, kad lėšos nebus prarastos dėl prasto transakcijų valdymo.

Transakcijos privalo turėti kelis požymius, kad būtų užtikrintas duomenų bazės vientisumas apdorojant vienu metu vykdomas transakcijas:

Atomiškumas (*atomicity*) Transakcijos kartais vadinamos **atomiškomis**, nes jos turi būti atliekamos kaip nedalomas vienetas. Transakcijos yra vienas ir nedalomas darbo vienetas – pavyksta arba visos, arbe nė viena.

Stabilumas (*durability*) Duomenų bazė turi būti pastovios būsenos. Baigus transakciją duomenų bazės būsena tampa patvaria ir ši būsena privalo tokia išlikti net įvykus sistemos klaidai.

Nuoseklumas (*consistency*) Apibūdina vienu metu kelių vartotojų vykdomų transakcijų rezultatus. Transakcijos negali būti **kaitomos** ar vykdomos dalimis kartu su kitomis transakcijomis, bet turi būti vykdomos nustatyta serijų tvarka.

Izoliavimas (*isolation*) Vienos transakcijos metu naudojami duomenys negali būti naudojami kitoje transakcijoje, kol bus baigta pirmoji transakcija.

Visada užtikrinamas vienam vartotojui skirtų duomenų bazių nuoseklumas ir izoliavimas, nes esant tik vienam vartotojui transakcijos visada bus vykdomos po vieną. Jeigu yra keli vartotojai, turi būti naudojamas DBVS valdymas ir užtikrinama, kad šie keturi transakcijų požymiai yra išsaugomi. Vienu metu vykdant transakcijas problemų gali kilti dėl trijų priežasčių. Gali skirtis terminija, bet iš esmės jos apibūdina tas pačias duomenų bazių vientisumo problemas. Mes vartosime 2000 m. įsigaliojusią terminiją, bet prireikus paminėsime ir kitą terminiją.

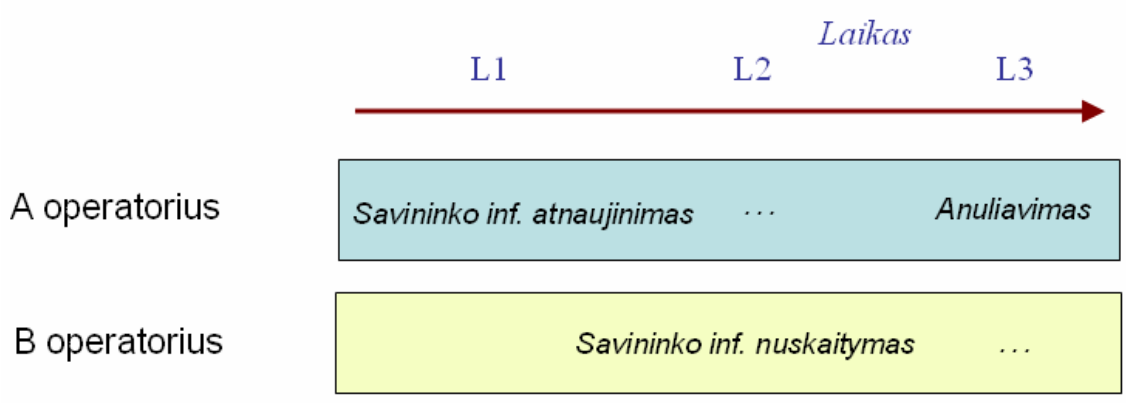
Prarasto atnaujinimo problema gali kilti tada, kai du atskiri vartotojai turi prieigą prie tų pačių duomenų bazės įrašų ir juos keičia. Pavyzdžiui, du GIS operatoriai atnaujina žemės sklypų duomenų bazės savininkus. Jeigu jie abu redaguotų sklypo savininko informaciją, rezultatas galėtų būti toks, kaip parodyta 3-1 pav.

3-1 pav. Prarasto atnaujinimo problema

Šiame paveikslėlyje kairėje pusėje parodyti du operatoriai ir laikas, skaičiuojamas iš kairės į dešinę. L1 laiku A operatorius nuskaito tam tikro žemės sklypo savininko informaciją. Tarkime, kad duomenų bazė pateikia vardą Smith. L2 laiku B operatorius nuskaito tuos pačius duomenų bazės duomenis ir gauna tokį patį atsakymą. L3 laiku A operatorius atnaujina sklypo informaciją pakeisdamas savininko vardą į Jones. L4 laiku B operatorius atnaujina savininko vardą į Thomas. Šio operacijų rinkinio rezultatas yra sklypo savininko nustatymas į Thomas. Vertė Jones, kurią įvedė A operatorius L3 laiku buvo prarasta. L3 laiku įvesta vertė buvo įterpta tarp B operatoriaus L2 laiku gautos informacijos ir L4 laiku įvykdyto atnaujinimo, todėl B operatorius net nematė, kad L3 laiku buvo įvestą vertė ir L4 laiku įvedė savo vertę.

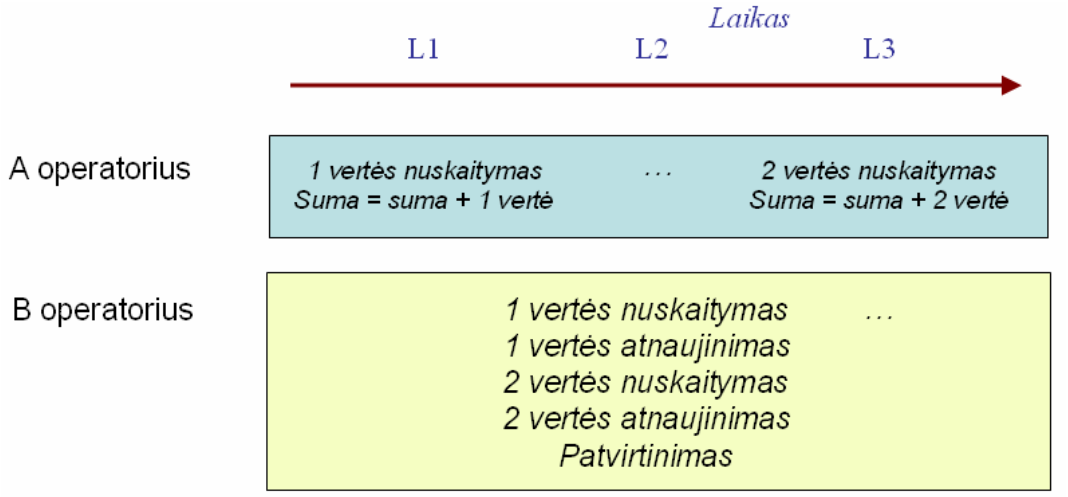
3-1 pav. Prarasto atnaujinimo problema

Nesusietoji priklausomybės problema (dar vadinama laikinąja atnaujinimo problema) gali kilti kai vienos transakcijos metu atnaujinamas duomenų bazės elementas, bet tada įvyksta transakcijos klaida ir atnaujinimas anuliuojamas. 3-2 pav. pavaizduota tokia situacija. Darome prielaida, kad sklypo savininkas yra Smith. A operatorius L1 laiku pakeičia sklypo savininką į Jones. L2 laiku B operatorius nuskaito savininko informaciją ir gauna vertę Jones. L3 laiku įvyksta A operatoriaus transakcijos problema, anuliuojama visa transakcija, o savininko vertė (informacija apie savininką) atstatoma į Smith. B operatorius vis dar dirba galvodamas, kad savininkas yra Jones, nors ši vertė duomenų bazėje niekada nebuvo pastovi.



3-2 pav. Nepatvirtintos priklausomybės problema

Nenuoseklios analizės problema (dar vadinama neteisingos suvestinės problema) gali atsirasti, jeigu transakcija apskaičiuoja suvestinės vertes, pvz., sumą ar vidurkį, kol kitos transakcijos atnaujina vertes. Tokiu atveju, suvestinės transakcija gali nuskaityti tam tikras vertes prieš jas pakeičiant, o kitas vertes įvykdžius transakciją. 3-3 pav. pavaizduota tokia situacija. Tarkime, kad du operatoriai apdoroja dviejų žemės sklypų informaciją, pirmojo vertė yra 50 000 JAV dol., o antrojo – 70 000 JAV dol. A operatorius L1 laiku pradeda nustatyti bendrąją abiejų sklypų vertę nuskaitydamas pirmojo sklypo vertę. Duomenų bazė pateikia 50 000 JAV dol. vertę. A operatoriaus transakcija prideda šią vertę (50 000 JAV dol.) prie kintamojo *sumos* dydžio, kuriuo bus „pateikta“ apskaičiuota suma. L2 laiku B operatorius atlieka kelis pakeitimus. Tai pirmojo sklypo vertę pakeičia į 60 000 JAV dol., o antrojo sklypo vertę į 80 000 JAV dol. B operatorius patvirtina abu šiuos atnaujinimus. L3 laiku A operatorius toliau skaičiuoja sumą nuskaitydamas antrojo sklypo vertę. Duomenų bazė pateikia 80 000 JAV dol. Ši suma pridedama prie esamo kintamojo sumos dydžio ir pateikia bendrą 130 000 JAV dol. dviejų sklypų vertę.



3-3 pav. Nenuoseklios analizės problema

Problema ta, kad dviejų sklypų verčių suma prieš B operatoriui pakeičiant vertes buvo 120 000 JAV dol. B operatoriui atlikus visus reikiamus pakeitimus, bendra dviejų sklypų vertė buvo 140 000 JAV dol. A operatorius apskaičiavo 130 000 JAV dol. vertę, kuri nėra nei senoji, nei naujoji vertė. Ji yra prieštaraujanti suma, kurią skaičiuojant buvo naudojama viena senoji suma ir viena naujoji suma.

Visais paminėtais atvejais problemos kyla tik todėl, kad dvejoms ar daugiau transakcijų reikia prieigos prie tos pačios duomenų bazės informacijos ir bent viena transakcija vykdo „rašymo“ operaciją. Dvi vienu metu vykstančios „skaitymo“ operacijos nesukelia duomenų bazės nesuderinamumo.

3.1.2 Išteklių blokavimas

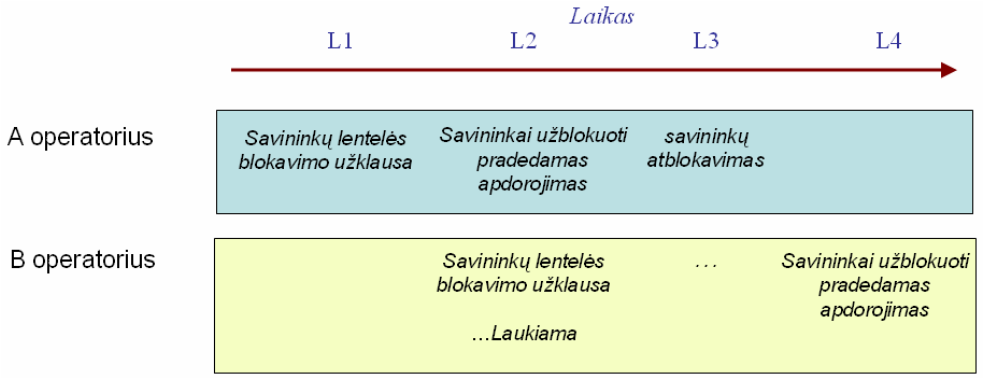
Visos trys problemos kyla dėl tuo pačiu metu vykdomos prieigos prie duomenų bazės. Izoliavus transakcijas jos būtų teisingos, bet įsiterpus kitoms transakcijoms jos gali pateikti neteisingas vertes. Vienu metu vykdomų apdorojimų valdymo būdas vadinamas **blokavimu** *(locking)*. Pagrindinė prielaida yra ta, kad transakcijai reikia išimtinio išteklių naudojimo, pvz., duomenų bazės įrašo. Transakcija blokuoja visas kitas transakcijas, kad jos negalėtų pakeisti išteklių, kol bus baigta blokuojanti transakcija. Visos paskesnės transakcijos, norinčios gauti prieigą prie tų pačių duomenų bazės išteklių, turi palaukti kol nebebus blokuojami tie ištekliai. Tokiu atveju transakcija gali atlikti visus reikiamus veiksmus ir tol, kol bus panaikintas blokavimas, kitos transakcijos negalės pakeisti reikiamo objekto. Blokavimai galimi keliais duomenų bazės lygiais: duomenų bazės, lentelės, puslapio, eilutės ar stulpelio. Blokavimo lygis arba blokuojamo objekto dydis vadinamas **blokavimo skaidymo gyliu** (lock granularity).

Duomenų bazės blokavimai

Duomenų bazės lygiu blokuojama visa duomenų bazė ir neleidžiama naudoti jokių lentelių, kol baigiama blokuojanti transakcija. Tai nėra įprastas metodas ir jį galima naudoti tik vieno vartotojo naudojamoje duomenų bazėje arba vykdant didelės apimties paketinius procesus.

Lentelių blokavimai

Lentelių lygiu visa lentelė blokuojama tol, kol tam tikra transakcija atlieka veiksmą. Daugeliu atvejų pakeitimai atliekami keliose lentelėse ir blokuojamos visos reikiamos lentelės. Dvi transakcijos gali būti vykdomos toje pačioje duomenų bazėje, jeigu jos turi prieigą prie skirtingų lentelių. 3-4 pav. pavaizduotas blokavimas lentelių lygiu. Šiame pavyzdyje matome, kad A operatoriui pradėjus transakciją *savininkų* lentelė užblokuojama ir kitos transakcijos negali pakeisti joje esančių duomenų. L2 laiku B operatorius pradeda transakciją, kurios metu reikia pakeisti *savininkų* lentelės duomenis, bet dėl A operatoriaus blokavimo turi palaukti, kol bus panaikintas blokavimas. L3 laiku baigiama A operatoriaus transakcija ir panaikinamas *savininkų* lentelės blokavimas. L4 laiku B operatoriui suteikiama *savininkų* lentelės blokavimo teisė ir jis gali pradėti šios lentelės operacijas. Taip uždraudžiamos kelių transakcijų prieigos prie *savininkų* lentelės.



3-4 pav. Lentelės blokavimas

Puslapio blokavimai — Puslapio lygio blokavimai apriboja prieigą prie pačios saugojimo laikmenos. Puslapis arba disko puslapis yra disko dalis. Puslapiai būna fiksuoto dydžio, pvz., 8 K, 16 K, 32 K ir t. t. Tam tikras puslapis yra saugojimo disko dalis, todėl, viename puslapyje gali būti vienos lentelės dalis, visa lentelė ar kelių lentelių dalys. Jei transakcijai reikia blokuoti duomenų bazės objektą, blokuojamas visas laikmenos puslapis, kuriame gali būti kelių lentelių duomenys arba tik nedidelė vienos lentelės dalis, atsižvelgiant į kiekvienos lentelės dydį ir kiekvienos lentelės paskirstymą laikmenos puslapiuose. Puslapio lygio blokavimas yra vienas dažniausių DBVS blokavimo būdų.

Eilučių blokavimai — Eilučių lygio blokavimas yra daug mažiau apribojantis, nes leidžia kitų transakcijų prieigą prie tos pačios lentelės, bet ne prie užblokuotos eilutės. Šis būdas labai padidina duomenų prieinamumą, nes blokuojamos tik tam tikros eilutės. Tačiau gali būti sudėtinga valdyti daug eilučių lygio blokavimų.

Laukų blokavimai — Laukų lygio blokavimai leidžia vienu metu vykdomų transakcijų prieigą prie tos pačios eilutės kol transakcijos naudoja skirtinguose tos pačios lentelės laukuose esančius duomenis. Kaip ir eilučių lygio blokavimų atveju, visų duomenų bazės laukų blokavimų tvarkymas nusveria transakcijų prieigos lankstumo teikiamą naudą.

Skiriasi ir blokavimų tipai. Tam tikro objekto (lentelės ar eilutės) blokavimo būsena gali būti šių trijų tipų.

Neužblokuota — Kitos transakcijos objekto nenaudoja, todėl galima užblokuoti prieigą.

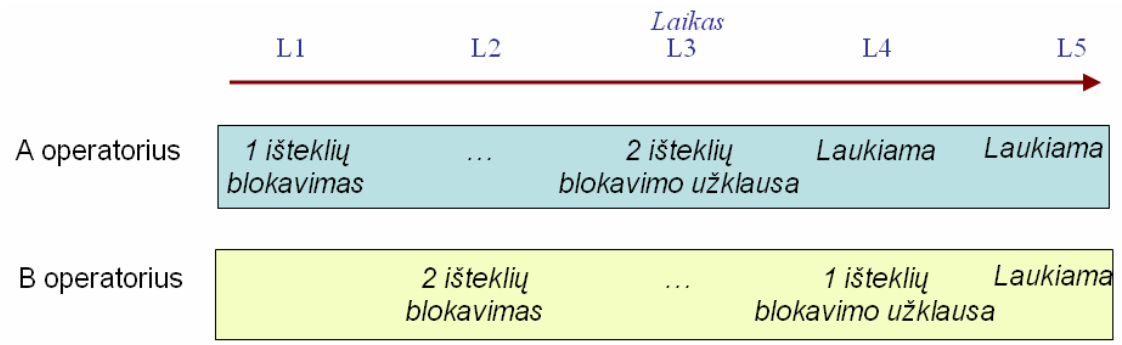
Bendras blokavimas — Bendras blokavimas yra „skaitymo“ blokavimas. Tai reiškia, kad transakcija nuskaito objektą. Todėl kitos transakcijos gali nuskaityti tą patį objektą tuo pačiu metu be nepageidautinų pasekmių, bet jokiai transakcijai negali būti suteikta teisė keisti objektą. Kelioms transakcijoms, kurioms reikia nuskaitymo prieigos prie objekto, gali būti suteikta bendro objekto blokavimo teisė. Transakcijos,

kurioms reikia keisti objektą, turi palaukti, kol bus panaikintas bendras blokavimas.

Išimtinis blokavimas — Išimtinis blokavimas yra „rašymo“ blokavimas. Jei transakcija pakeičia duomenų bazės vertę, tas objektas turi būti užblokuojamas naudojant išskirtinį blokavimą, kad kitos transakcijos neturėtų jokios prieigos prie objekto. Kaip matome pirmiau pateiktose problemose, tuo pačiu metu vykdomų transakcijų problemas sukelia „rašymo“ prieiga.

3.1.3 Abipusė blokuotė

Nors blokavimas išsprendžia tuo pačiu metu vykdomos prieigos prie duomenų bazės problemas, jis gali kelti naują problemą. **Abipusė blokuotė** yra padėtis, kai dvi ar daugiau transakcijų yra tokios pačios laukimo būsenos ir kiekviena laukia, kol kita panaikins išteklių blokavimą. 3-5 pav. parodyta abipusę blokuotę sukeliančios situacijos pavyzdys.



3-5 pav. Abipusė blokuotė

Šiame pavyzdyje A operatorius pradeda transakciją, kuriai reikia išimtinių išteklių, pažymėtų kaip „1 ištekiai“) blokavimo. L2 laiku B operatorius pradeda transakciją, kuriai reikia išimtinio antrų išteklių blokavimo. L3 laiku A operatoriaus transakcija pareikalauja 2 išteklių blokavimo. B operatorius jau užblokavo šiuo išteklius, todėl A operatoriaus transakcija turi palaukti, kol bus panaikintas blokavimas. L4 laiku B operatoriaus transakcija pareikalauja 1 išteklių blokavimo. Šie ištekliai jau išimtinai užblokuoti, todėl B operatoriaus transakcija irgi tu laukti. L5 laiku abi transakcijos laukia, kol kita panaikins blokavimą ir abi jos negali būti tęsiamos.

Vienas abipusių blokuočių valdymo būdas yra užtikrinimas, kad nekils abipusės blokuotės situacijų. Pagrindinis būdas tai atlikti yra „priversti“ transakciją gauti visus blokavimus prieš vykdant transakciją. Naudojant šį būdą išvengiama transakcijų konfliktų, nes blokai blokuojami vienas po kito. Šis būdas efektyvus, bet nesumažina vienu metu atliekamų transakcijų, nes užblokuoja daug išteklių vienu metu. Antras būdas išvengti abipusių blokuočių yra nustatyti kiekvieno duomenų bazės elemento tvarką ir užtikrinti, kad transakcijos, kurioms reikia atlikti kelis blokavimus, blokuoja elementus pagal nustatytą tvarką. Pavyzdžiui, naudojant lentelių blokavimo sistemą, kiekvienai lentelei priskiriama vieta eilėje ir kelios lentelės bus blokuojamos pagal iš anksto nustatytą tvarką. Blokuodami lenteles nustatyta tvarka mes pašaliname abipusės blokuotės galimybę. Tačiau šį schema reikalauja, kad programuotojai ir pati DBVS žinotų ir tiksliai vykdytų priskirtą išteklių tvarką.

Yra dar keli abipusių blokuočių išvengimo būdai, bet visų jų požymiai panašūs. Dėl transakcijų blokavimų apribojimų naudojant visus būdus smarkiai sumažėja našumas ir apribojamos vienu metu vykdomos transakcijos.

Alternatyvus abipusių blokuočių valdymo būdas yra kuo efektyviau blokuoti prieigą (bet leisti abipuses blokuotes) ir nustatyti abipusių blokuočių būseną, kai tokia atsiranda. Nustačius tokia būseną viena iš abipusės blokuotės transakcijų yra nutraukiama ir paleidžiama iš naujo, o kita (-os) abipusės blokuotės transakcija (-os) užbaigiama (-os). Šis būdas ypač naudingas kai transakcijos dažniausiai būna trumpos ir mažai tikėtina, kad vienu metu vykdomoms transakcijoms reikės prieigos prie tų pačių duomenų bazės elementų.

Paprastas būdas, kurį naudodamos DBVS aptinka abipusę blokuotę yra transakcijų stebėjimas. Bet kuriai transakcijai, kuri per tam tikrą iš anksto nustatytą laikotarpį neįvykdo užduoties, priskiriama abipusės blokuotės būsena. Sudėtingesnis abipusių blokuočių nustatymo būdas yra **laukimo diagramos** (*wait-for-graph*) valdymas. Šioje diagramoje saugomas transakcijų T_i ir T_j ryšys. Atsižvelgiama, ar transakcija T_i laukia, kol T_j panaikins išteklių blokavimą. Jeigu diagramoje egzistuoja ciklas, vadinasi egzistuoja abipusės blokuotės būsena, todėl reikia nutraukti vieną iš ciklo transakcijų ir taip baigti abipusę blokuotę.

Nustačius abipusę blokuotę pasirenkama **nukentėsianti** transakcija. Yra daug nukentėsiančių transakcijų parinkimo algoritmų, bet nepageidautina parinkti tą transakciją, kuri apima daug operacijų ir yra seniai paleista. Pats efektyviausias nukentėsiančios transakcijos pasirinkimas būtų ką tik pradėta vykdyti trumpa transakcija. Pasirinkus nukentėsiančią transakciją, ji nutraukiama ir atkuriama. Tam tikros DBVS iš naujo paleidžia nukentėjusias transakcijas darydamos prielaidą, kad abipusę blokuotę sukėlusios sąlygos bus pasikeitusios ir antrą kartą transakcija bus baigta sėkmingai. Kitos sistemos pateikia išimties kodą, kuris informuoja vartotoją ar vykdančią programą, kad transakcija nukentėjo dėl abipusės blokuotės ir nebuvo įvykdyta. Tokios sistemos pasikliauja, kad vartotojai ar programos susitvarkys nustatytu būdu.

3.1.4 Vienalaikiškumas GIS sistemoje

Daugeliu atveju duomenų bazės transakcijos užbaigiamos labai greitai. Oro linijų bilietų rezervavimo sistemos ar bankininkystės taikomosios programos redagavimai gali būti baigti per kelias sekundes. Tokios transakcijos vadinamos **trumpomis transakcijomis**. Tokiais atvejais mūsų aptarti blokavimo būdai yra efektyvūs ir užtikrina, kad vartotojai nuolat mato pastovią duomenų bazę, pakeitimai atliekami gerai apgalvota tvarka. Dauguma GIS operacijų, pvz., žemės sklypo vardo keitimas, irgi yra trumpos transakcijos ir gali sėkmingai naudoti pirmiau aptartus vienu metu vykdomų transakcijų valdymo būdus.

Tačiau GIS sistemoje yra daug transakcijų, kurios užtrunka žymiai ilgiau ir apima daug tarpusavyje susijusių įrašų. Erdvinių duomenų redagavimas gali būti sudėtingas. Pavyzdžiui, kuriant naują žemės sklypo planą ar rengiant kasyklos panaudojimo planą reikia redaguoti daug erdvinių elementų klasių, kurios gali būti susijusios tarpusavyje. Tokios operacijos vadinamos ilgomis transakcijomis ir gali trukti ilgiau negu kelias valandas ar net kelias savaites. Jas gali sudaryti kelios trumpos transakcijos, bet jos

apjungia darnių operacijų rinkinį į vieną transakciją. Ilgų transakcijų negalima paveikti įprastais išteklių blokavimo būdais, nes jos ilgam susieja didelius geografiniu plotus su daugeliu elementų klasių.

Ankstesnės GIS programos buvo pagrįstos CAD sistemomis, kai erdviniai duomenys galėjo būti saugomi kaip žemėlapio lapų brėžinių serijos. Tokiu atveju vartotojai galėjo išimtinai naudoti vieną žemėlapio lapą visos transakcijos metu. Nebuvo būtinybės valdyti vienu metu vykdomas transakcijas, nes kiekvienas lapas funkciškai buvo vienam vartotojui skirta duomenų bazė. Trūkumas tas, kad sudėtinga tvarkyti objektą, kuris apima daugiau negu vieną žemėlapio lapą.

Taikomosios GIS programos vystėsi į komercines RDBVS ir atsirado galimybių saugoti dideles vientisas duomenų bazes. RDBVS teikė nemažas efektyvumo, saugos ir duomenų vientisumo galimybes, bet negalėjo būti vykdomos ilgos transakcijos, kurios dažnai naudojamos daugelyje taikomųjų GIS programų. Tam tikri erdvinių duomenų formatai, pvz., ESRI SHAPE failas ar *Coverage* sluoksnis, funkciškai naudoja lentelių lygio blokavimą, nes vienu metu tik vienas vartotojas gali redaguoti konkrečią elementų klasę.

Kiti GIS teikėjai pasirinko vykdyti ilgas transakcijas naudodami išregistravimą / užregistravimą (*Check-Out/Check-In*). Šis būdas suteikia vartotojams galimybę nustatyti erdvinę apimtį ir išregistruoti vieną ar daugiau erdvinių elementų klasių ir taip sukurti asmeninę jų kopiją. Išregistravę kopiją vartotojai gali atnaujinti žemės sklypo planą ar iš naujo projektuoti kelių tinklą vienam vartotojui skirtu režimu. Kiti vartotojai neturi prieigos prie šios srities išregistruotų erdvinių elementų klasių. Baigus redaguoti užregistruojami pakeitimai ir atnaujinama pagrindinė duomenų bazė. Funkciškai šis būdas yra žemėlapio lapų blokavimo ir eilučių lygio blokavimų derinys.

Išregistravimo / užregistravimo pranašumas yra tas, kad išregistruotas duomenų bazes galima laikyti vietiniuose kompiuteriuose ir sumažinti serverio apdorojamų duomenų kiekį. Vienintelė užduotis, kurias reikia atlikti serveriui yra išregistravimo ir užregistravimo apdorojimas. Šio būdo trūkumai yra tokie, kad išregistravimo ir užregistravimo procesai gali ilgai trukti ir gali būti sudėtinga valdyti erdvinių elementų klasių ryšius bei užtikrinti, kad visi susiję erdviniai elementai yra tinkamai išregistruoti. Be to, naudojant šį būdą vartotojai gali ilgam netekti prieigos prie didelių geografinių duomenų sričių.

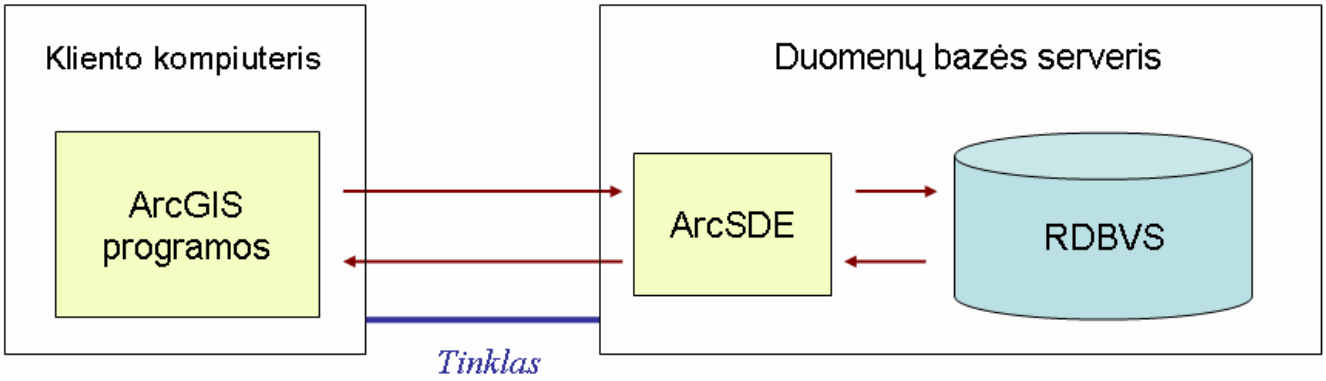
Alternatyvus kitų pasirinktas būdas (*Smallworld, ArcSDE*) dažnai vadinamas versijų sistema. Naudojant versijų sistemą vienu metu vykdomos transakcijos valdomos atskiriant kiekvieną transakciją ir suderinant ją užbaigus. Nustatomi konfliktai ir sprendžiant problemas gali prireikti vartotojo įsikišimo. Tolesnėse temose aptarsime vienu metu vykdomų transakcijų valdymo būdą *ArcSDE*.

3.2 ESRI keliems vartotojams skirta geoduomenų bazė

Šioje temoje nagrinėjamas ESRI kelių vartotojų duomenų bazės naudojimas. *ArcGIS* taikomųjų programų paketas realizuoja vienalaikiškumą panaudodamas programinę įrangą, pavadintą *ArcSDE* (*Arc Spatial Database Engine*), kuri naudoja metodą, pavadintą versijų sistema.

3.2.1 Kas yra SDE?

2 dalyje pristatėme *ArcSDE* geoduomenų bazes ir teigėme, kad jos yra geoduomenų bazių, saugančių informaciją pramonės standartus atitinkančioje RDBVS, pvz., DB2, *Oracle*, SQL *Server* ar *Informix*, tipas. Žodžių junginiu „*ArcSDE* geoduomenų bazė“ pasakoma, kad naudojami duomenys saugomi *ArcSDE* programoje, kas nėra tiesa. *ArcSDE* yra programinė įranga. Duomenys joje nesaugomi ir ji nenaudojama jiems valdyti. Ji leidžia mūsų *ArcGIS* taikomosioms programoms, pvz., *ArcMap* ir *ArcCatalog*, keistis duomenimis su RDBVS, kurioje fiziškai saugomi GIS duomenys. 3-6 pav. parodytas paprastas *ArcSDE* funkcijų pavyzdys.



3-6 pav. *ArcSDE* sąveika

Kai kliento taikomajai programai, pvz., *ArcMap* prireikia prieigos prie duomenų, saugomų SDE geoduomenų bazėje, ji keičiasi informacija su SDE. SDE konvertuoja šią užklausą į reikiamą SQL, skirtą naudojamai RDBVS (pvz., *Oracle*) ir persiunčia užklausą į RDBVS. SDE paima RDBVS pateiktus duomenis ir perduoda juos kliento taikomajai programai, kur jie yra rodomi ir analizuojami. Naudodama taikomąsias GIS programas rodymo ir analizavimo funkcijoms valdyti ir RDBVS saugojimo, nuskaitymo ir priežiūros funkcijoms valdyti, *ArcSDE* naudoja stipriąsias abiejų programinės įrangos programų puses.

ArcSDE yra informacijos tarp *ArcGIS* ir erdvinių duomenų bazės perdavimo kanalas, todėl ji tvarko palaikomos RDBVS skirtumus. Tarp taikomųjų šių abiejų duomenų bazių serverių programų gali būti nemažai skirtumų, įskaitant duomenų tipus, erdvinių duomenų tipų palaikymą, indeksavimą ir duomenų bazės apribojimų taikymą. SDE užtikrina, kad nepaisant to, kur saugomi erdviniai duomenys (*Oracle* ar DB2), visada bus užtikrinama nuosekli duomenų sąsaja. GIS taikomoji programa, sukurta prieigai prie SQL serveryje saugomų duomenų naudojant *ArcSDE*, veiks taip pat *Oracle*, DB2 ar *Informix* sistemose.

Duomenys saugomi galingoje trečiosios šalies RDBVS aplinkoje, todėl galime pasinaudoti RDBVS pranašumu lyginant su failų formatais, pvz., *Coverage* sluoksniais ar su mažiau sudėtingomis asmeninėmis ar failų geoduomenų bazėmis. Didžiausias pranašumas yra duomenų apsauga. RDBVS valdo įprastas duomenų priežiūros funkcijas, pvz., atsarginių kopijų kūrimą ir duomenų atkūrimą, todėl duomenys yra daug patikimiau apsaugoti. Be to, galima saugoti didelius duomenų kiekius ir efektyviai pateikti juos GIS vartotojui. Kaip aptarėme ankstesnėse temose, galime tvarkyti daug vienalaikių vartotojų.

ArcSDE yra trijų lygių.

Asmenims skirta

ArcSDE — Pagrindinis *ArcSDE* realizavimas įtrauktas į *ArcGIS*. Ji veikia *SQL Server Express* RDBVS aplinkoje, kuri teikiama nemokamai. *SQL Server Express* turi kelis apribojimus, įskaitant duomenų bazės ribojimą iki 4 GB. Asmenims skirta *ArcSDE* nepalaiko vienu metu vykdomų transakcijų. Vienu metu gali redaguoti tik vienas asmuo, bet palaiko nedidelį skaičių vienalaikių vartotojų, turinčių prieigą tik skaityti. Vartotojai gali tvarkyti *Personal ArcSDE* naudodami *ArcCatalog*, todėl nereikia papildomų RDBVS tvarkymo įgūdžių ar patirties.

Darbo grupėms skirta

ArcSDE — Kaip ir asmenims skirta *ArcSDE*, darbo grupėms skirta *ArcSDE* veikia laisvai platinamos *SQL Server Express* versijos aplinkoje ir ją galimą tvarkyti naudojant *ArcCatalog*. Jai irgi taikomas 4 GB duomenų bazės apribojimas, bet šio lygio *ArcSDE* palaiko iki 10 tuo pačiu metu sistemą naudojančių vartotojų.

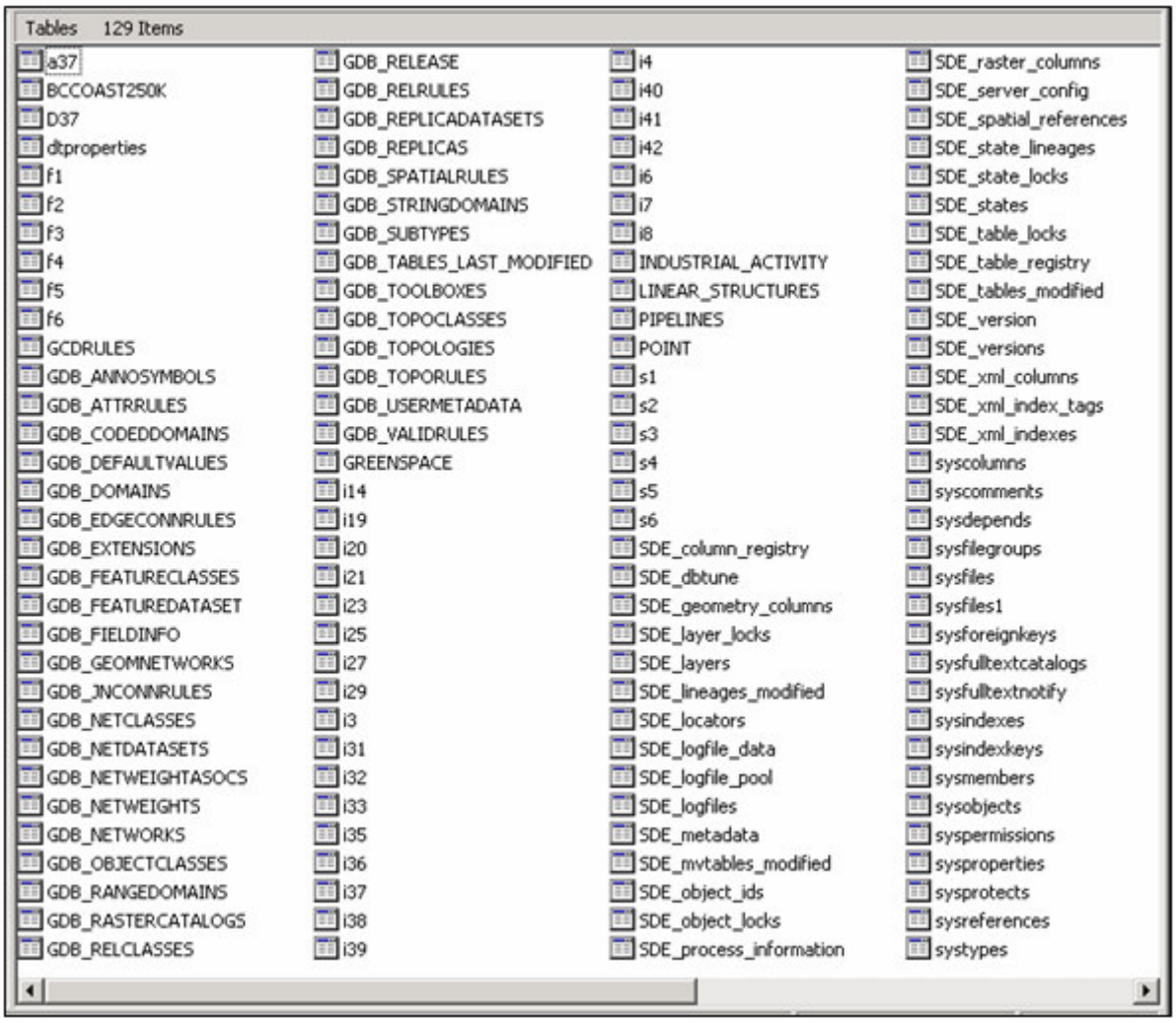
Įmonės skirta *ArcSDE* — Tai yra visas funkcijas turinti *ArcSDE* programa, veikianti *Oracle*, *SQL Server*, DB2 ir *Informix* sistemose. Šio lygio SDE palaiko bet kokio dydžio duomenų bazes ir bet kurį vienu metu sistemą naudojančių vartotojų skaičių. Naudodami įmonės skirtą *ArcSDE* vartotojai naudoja nuosavą RDBVS licenciją ir turi valdyti bei tvarkyti duomenų bazę naudodami su RDBVS pateiktus įrankius. Įmonėms skirtas *ArcSDE* tvarkyti dažnai reikia duomenų bazės administratoriaus (DBA). Dėl papildomų RDBVS ir panašių sistemų, kurių geoduomenų bazėms tvarkyti reikia DBA, kaštų, įmonėms skirta SDE gali būti ženkliai brangesnė negu *SQL Server Express* parinktys.

Paskesnėse temose nagrinėsime vienalaikiškumą, todėl jos bus susijusios tik su darbo grupėms ir įmonėms skirtų sistemų realizavimu.

ArcSDE prie geoduomenų bazės prideda kelias lenteles. SQL serveryje saugomoje SDE geoduomenų bazėje yra vartotojo nustatytų lentelių, kuriose atvaizduojamos erdvinių elementų klasės ir neerdvinių duomenų lentelės, geoduomenų bazės lentelių, kurios naudojamos valdant geoduomenų bazės sudėtines dalis, pvz., domenus ir potipius, SDE lentelių, kurios naudojamos valdant SDE ir versijų sistemos funkcijas ir lentelių, kurių reikia pačiam SQL serveriui, kad būtų galima valdyti vartotojus, teises ir rodykles.

3-7 pav. pavaizduotos nedidelėje SDE geoduomenų bazėje esančios lentelės, kurios saugomos SQL serveryje. Lentelės, pvz., *INDUSTRIAL_ACTIVITY* ir *PIPELINES*, yra vartotojo nustatomos lentelės, kurios naudojamos dviem erdvinių elementų klasėms apibūdinti. Lentelės su priešdėliu *GDB_*, pvz., *GDB_Domains*, naudojamos valdyti geoduomenų bazės sudėtines dalis. Lenteles su priešdėliu *SDE_* naudoja tik SDE, o lentelės su priešdėliu *sys* yra vidinės lentelės, kurias naudoja SQL serveris. Be to, lentelės su labai trumpais ir sunkiai suprantamais pavadinimais, pvz., f1, d37 ir t. t. yra vidinės SDE lentelės. Šios lentelės atlieka nemažai funkcijų:

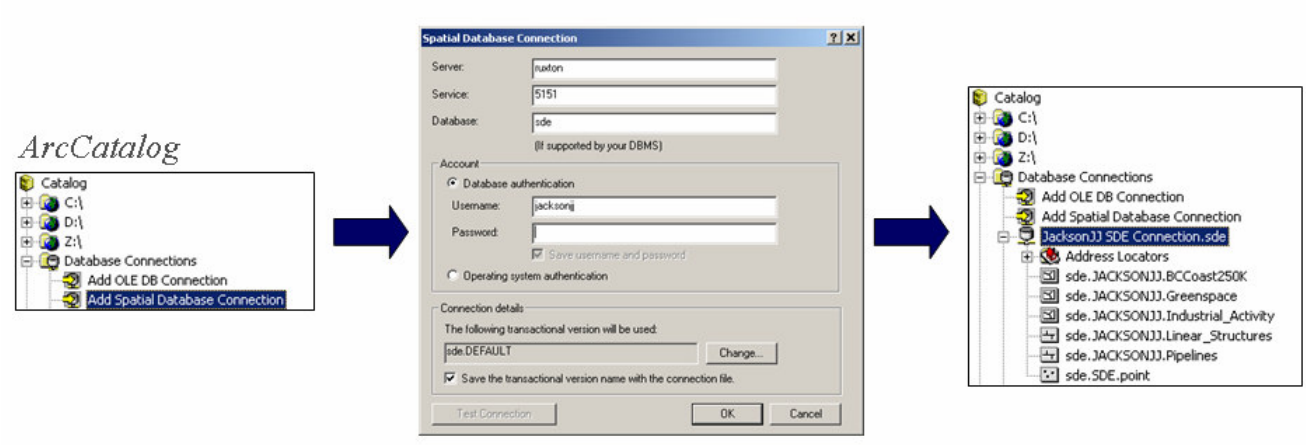
Elementų lentelės	Turinčios priešdėlį „f“. Šiose lentelėse saugomi elementų geometrijos ir kiti elementų klasės metaduomenys, Kiekviena sunumeruota lentelė, pvz., f5, nurodo skirtingą elementų klasę.
Erdvinės rodyklės	Turinčios priešdėlį „s“. Jose saugoma erdvinių elementų klasės erdvinė rodyklė. Kaip ir erdvinių elementų lentelės, kiekviena lentelė taikoma skirtingai erdvinių elementų klasei.
Delta lentelės	Turinčios priešdėlį „A“ arba „d“. Šiose lentelėse saugomi kiekvienos erdvinių elementų klasės versijų sistemos papildymai ar panaikinimai.



3-7 pav. SDE geoduomenų bazės lentelės, esančios *SQL Server*

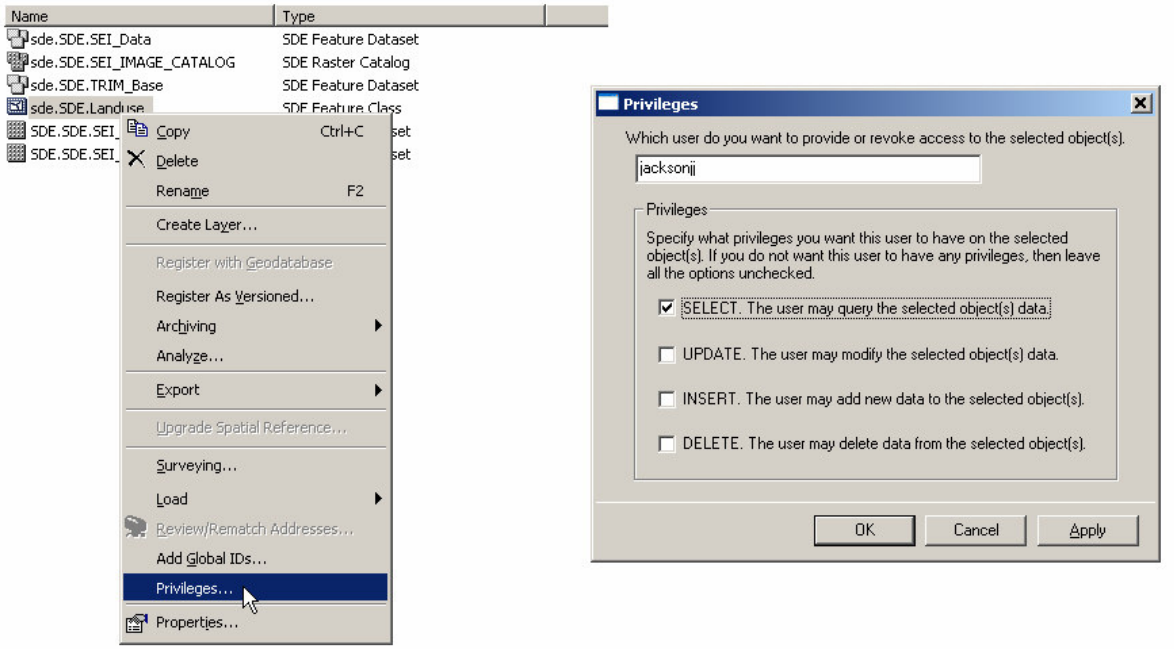
Šias lenteles kuria ir valdo SDE arba pati RDBVS, todėl vartotojai turi vengti geoduomenų bazės keitimų tiesiogiai naudojant DBVS. Pakeitus duomenis dali kilti problemų vienos ar daugiau geoduomenų bazės elementų klasių problemų.

ArcSDE geoduomenų bazės apsaugą valdo RDBVS, todėl norėdami naudoti SDE geoduomenų bazę, vartotojai turi prisijungti prie duomenų bazės pateikdami vartotojo vardą ir slaptažodį. Arba prisijungti netiesiogiai naudodami tokį patį vartotojo vardą ir slaptažodį, kurį naudoja prisijungdami prie operacinės sistemos. Prisijungę prie duomenų bazės vartotojai gali sąveikauti su joje esančiomis elementų klasėmis ir objektais, kaip ir kitose geoduomenų bazėse. Prisijungiant prie SDE geoduomenų bazės galima naudoti *ArcCatalog*, kaip 3-8 pav.



3-8 pav. Prisijungimo prie duomenų bazės naudojant *ArcCatalog*

Kiekvienai SDE geoduomenų bazės elementų klasei gali būti suteiktos skirtingos prieigos teisės. Pagal numatytuosius nustatymus geoduomenų bazėje sukūrus naują elementų klasę vartotojas, sukūręs elementų klasę, turi visas teises (teisę skaityti ir teisę rašyti), o kiti geoduomenų bazės vartotojai teisių neturi. Kiti vartotojai negali net matyti naujų elementų klasių, nebent būtų suteiktos tokios teisės. Kiekvienam vartotojui, kuriam reikia prieigos prie elementų klasės teisių, elementų klasės kūrėjas turi suteikti reikiamas teises. 3-9 pav. parodyta kaip naudojant *ArcCatalog* galima suteikti tokias teises.



3-9 pav. Elementų klasės teisių nustatymas naudojant *ArcCatalog*

3.2.2 Versijų sistema

Aptarėme, kaip naudojant išteklių blokavimą valdomos trumpos vienu metu vykdomos transakcijos. Tačiau šis sprendimas neefektyvus naudojant ilgas transakcijas, kurios dažnai vykdomos GIS operacijose, nes reikėtų labai ilgam blokuoti daugelį išteklių. Alternatyvus ESRI parinktas būdas yra **versijų sistema**. Versijų sistema palaiko vienu metu vykdomas transakcijas sukurdama geoduomenų bazės versiją, kurioje saugomas

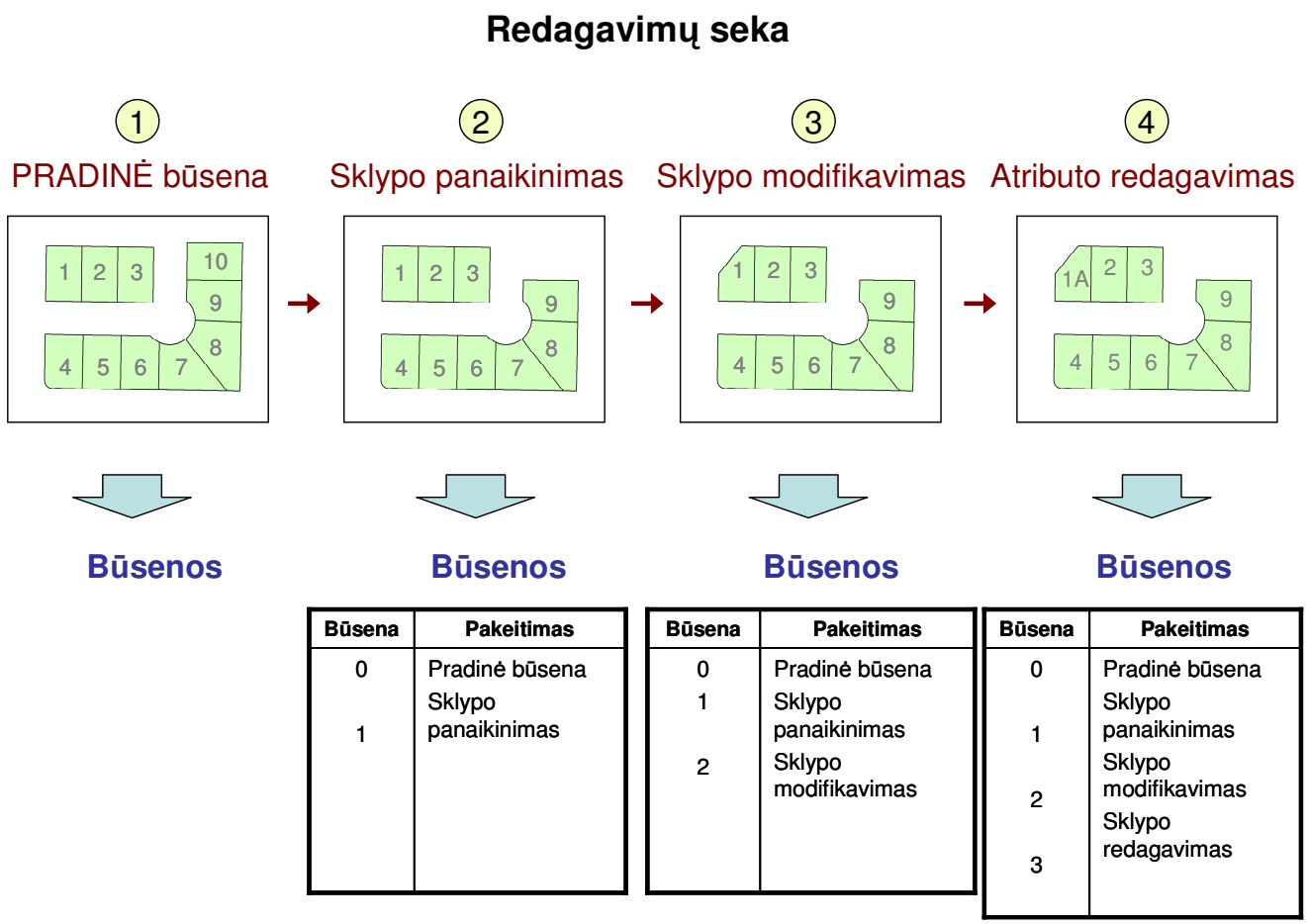
nepriklausomas duomenų bazės vaizdas. Ši versija įtraukia pakeitimus į duomenų bazę nekurdama atskirų duomenų kopijų. Vienu metu gali egzistuoti kelios versijos ir jos gali palaikyti kelis vartotojus.

Šis požiūris gali būti suprantamas kaip **optimistinis** vienu metu vykdomų transakcijų valdymo būdas. Išteklių blokavimas yra pesimistinis požiūris, nes juo numatoma, kad įvyks konfliktų ir atliekami duomenų vientisumo problemų valdymo veiksmai dar nekilus problemoms uždraudžiant antro vartotojo prieigą prie tam tikrų išteklių. Versijų sistema yra optimistinė, nes ji neblokuoja išteklių. Ji suteikia galimybę redaguoti konfliktus, bet neapriboja duomenų bazės dalių naudojimo, kol vykdomos ilgos transakcijos. Konfliktams redaguoti pateikiami įrankiai. Juos naudojant galima užbaigus transakciją rasti ir valdyti konfliktus.

Vietoje to, kad saugotų atskiras duomenų bazės kopijas, SDE išlaiko pradinius duomenis ir saugo kiekvieną pradinių duomenų pakeitimą. Pakeitimai valdomi naudojant taip vadinamas duomenų bazės **būsenas**. Būsenos yra atskiri momentiniai duomenų vaizdai po kiekvieno pakeitimo. Kiekviena redagavimo operacija, pvz., pridėjimas, panaikinimas ar modifikavimas sukuria naują būseną. SDE duomenų bazė turi pradinę (DEFAULT) versiją, kuri nurodo duomenų bazės būseną tuo metu, kai buvo pradėta taikyti duomenų versijų sistema. Taip sukuriama pradinė duomenų bazės būsena ir visi duomenų pakeitimai siejami su šia pradine (DEFAULT) būsena. Kiekvienas pakeitimas saugomas duomenų bazėje kaip būsena.

Kad galėtų valdyti duomenų bazės būsenas SDE tvarko daug lentelių, tarp jų ir būsenų (*States*) lentelę. Šiose lentelėse fiksuojami visi daromi pakeitimai ir saugoma laiko žymė, nurodanti pakeitimo laiką ir jį atlikusį vartotoją. Patys pakeitimai saugomi RDBVS sistemos lentelių rinkinyje, pavadintame **delta lentelėmis**. 3-7 pav. buvo parodytos duomenų bazės delta lentelės.

3-10 pav. parodytas sklypo duomenų bazės redagavimų rinkinys ir duomenų bazės būsenos. Kad būtų aiškiau, kiekvieną būsenos pakeitimą apibendrinama vienas Būsenų lentelės įrašas, bet tikrovėje apie kiekvieną būsenos pakeitimą saugoma informacija yra daug išsamesnė ir apima kelias lenteles.



3-10 pav. Būsenos redaguojant duomenų bazę

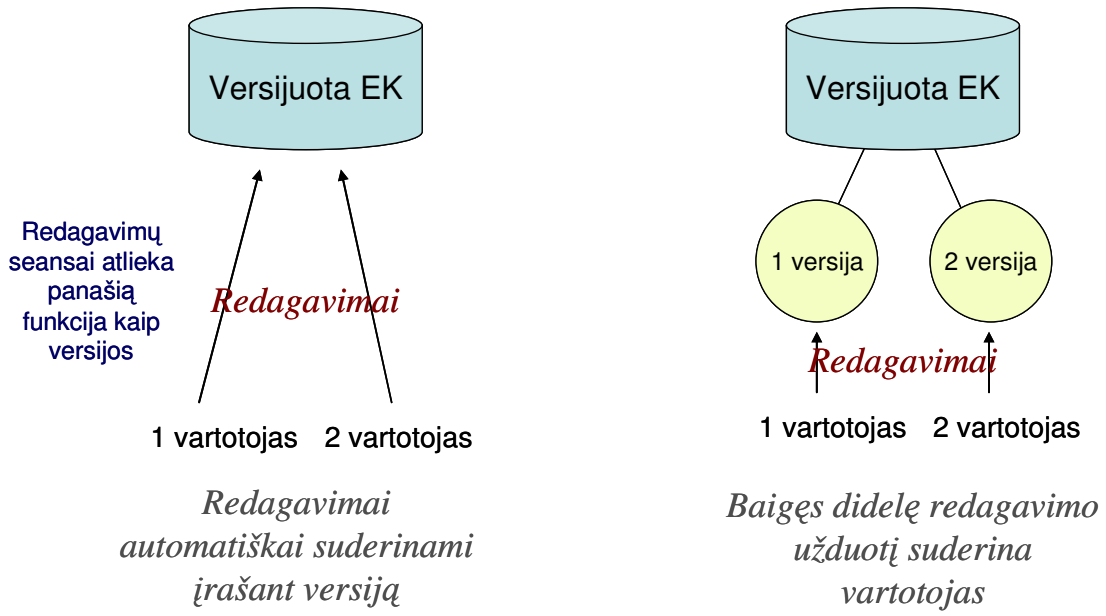
Viršutiniame kairiajame paveikslėlyje parodyta pradinė būsena ir 10 sklypų. Šios pradinės būsenos *Būsenų* lentelėje neturi būti nurodyti elementų pakeitimai, todėl nurodoma PRADINĖ būsena. 2, 3 ir 4 veiksmais atlikus pakeitimus, būsenos ir delta lentelėse pridedami kiekvieno pakeitimo įrašai. Atlikę tris pakeitimus turime tris būsenas ir pradinę būseną.

PRADINĖ versija ir viso šios versijos būsenos nurodo, kaip turi būti atvaizduota duomenų bazė. Kad *ArcMap* galėtų sudaryti versijos būsena, ji turi nusiųsti užklausą apie pradinę būseną (PRADINĘ versiją) ir apie kiekvieną kitą būseną, o tada sukurti atnaujintą duomenų bazės atvaizdavimą.

Jei norite naudoti versijų sistemą ir leisti keliems vartotojams redaguoti tam tikrą elementų klasę, ta elementų klasė turi būti *užregistruota versijų sistemai*. Registruojant sukuriamos reikiamos SDE duomenų bazės sistemos lentelės, kad pakeitimus būtų galima valdyti naudojant versijas. Registruojant paruošiama PRADINĖ versija. Vartotojai gali kurti naujas versijas pagal šią PRADINĘ versiją ir naudoti šias duomenų bazės versijas, tuo metu PRADINĖ versija lieka nepakeista ir yra matoma visiems vartotojams. Pavyzdžiui, kelių projektuotojas gali sukurti naują versiją ir pažymėti joje naują greitkelį. Suprojektavus naują versiją operatorius gali saugoti ją atskirai nuo PRADINĖS versijos kelias dienas ar savaites. Baigus projektą arba kelių tiesybą projektas gali būti pateiktas į PRADINĘ duomenų bazę, kad jį galėtų matyti ir naudoti visi vartotojai, perrašydami ankstesnės versijos atvaizdavimą. Versijos pakeitimų įrašymo į PRADINĘ versiją procesas vadinamas

versijos **skelbimu** *(posting)*. Versijos gali būti sukurtos pagal kitas versijas, todėl galima nagrinėti skirtingas projektavimo versijas nekeičiant pagrindinės duomenų bazės.

Vartotojas gali nesudėtingai naudoti versijų sistemą. Redaguojant erdvinius duomenis ArcSDE programa, elementų klasės funkcionuoja taip pat, neatsižvelgiant ar yra kuriamos jų versijos, ar ne. Naudojant elementų klasės versijų sistemą ir baigus redagavimą, skirtingų vartotojų pakeitimai bus suderinti (reconciled). Redagavimų derinimas yra neseniai baigtų redagavimų tikrinimas ir užtikrinimas, kad nėra konfliktų, kylančių dėl to, kad tuos pačius elementus redagavo kitas vartotojas. Atliekant trumpas transakcijas vartotojai gali redaguoti elementų klasę, kurios versijos buvo sukurtos, tiesiogiai pagrindinėje duomenų bazėje (PRADINĘ versiją). Atliekant ilgesnes transakcijas gali būti sukurta naują elementų klasės (3.11 pav. pavadinta „EK“) versija ir ji naudojama ilgesnį laikotarpį. Šie du požiūriai pavaizduoti 3-11 pav.



3-11 pav. Kelių vartotojų SDE redagavimas

Kairiajame paveikslėlyje vaizduojama trumpos transakcijos darbo eiga. Čia keli vartotojai tiesiogiai redaguoja PRADINĘ elementų klasės versiją. Kiekvienas vartotojas naudodamas redagavimo seansą nustato transakcijos trukmę. Redagavimo seansas atlieka panašią funkciją kaip versija. Jo metu redagavimai atskiriami nuo kitų vartotojų tol, kol baigiamas redagavimo seansas ir pakeitimai patvirtinami PRADINĖJE versijoje. Vartotojui įrašant redagavimus automatiškai atliekamas derinimas, kad būtų užtikrinta, jog nekilo konfliktų su kito vartotojo redagavimais.

3-11 pav. dešinėje pavaizduota situacija, kai du vartotojai sukuria dvi skirtingas darbo versijas. Tada vartotojai per kelias dienas ar dar ilgiau gali pradėti ir baigti redagavimo seansus, o PRADINĖ duomenų bazė taip ir nebūna keičiama. Įrašant redagavimus, pakeitimai įrašomi tik darbo versijoje. Baigus ilgą transakciją, kiekvienas vartotojas pareikalauja, kad visi versijų pakeitimai būtų suderinti su PRADINE versija.

Vartotojai gali sukurti savąsias elementų klasių, prie kurių jie turi prieigos teisę, versijas arba administratorius gali sukurti elementų klasės, prie kurios vartotojas neturi prieigos,

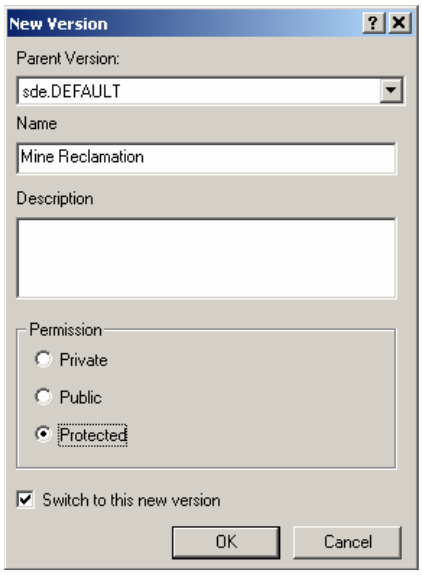
versiją. Kaip minėta pirmiau, versijos gali būti kuriamos pagal PRADINĘ versiją arba pagal kitas versijas. Kaip ir elementų klasėms, versijoms suteikiamos teisės, kuriomis apibrėžiama, kaip kiti vartotojai gali jas naudoti. Skirtingai negu elementų klasės, teisės nenustatomos konkretiems vartotojams. Jos nustatomos visiems vartotojams iškart. Gali būti nustatyta viena iš trijų naujos versijos teisės verčių:

Asmeninė (*private*) Nauja versija nepasiekiama kitiems vartotojams.

Apsaugota (*protected*) Visi vartotojai gali peržiūrėti versiją, bet negali jos keisti.

Vieša (*public*) Visi vartotojai turi visą prieigą (skaityti ir rašyti) prie versijos.

3-12 pav. parodytas naujos versijos kūrimo procesas ir versijos teisių suteikimas.



3-12 pav. Versijos kūrimas

Darbo eigos pavyzdžiai

Naudojant SDE ir versijas galimos kelios skirtingos darbo eigos. Tolesniame skyriuje parodytos kelios galimos įvykių sekos ir SDE panaudojimo būdas įgyvendinant darbo eigai keliamus reikalavimus.

1 scenarijus:

- Keli vartotojai tuo pačiu metu atlieka nesudėtingus duomenų bazės modifikavimus – įterpimas, atnaujinimas ir šalinimas.

Sprendimas:

- Keli vartotojai vykdo atskirus tos pačios PRADINĖS versijos redagavimo seansus. Jeigu vartotojai redaguoja tą patį elementą, tas, kuris savo pakeitimus įrašo antras yra priverstas juos suderinti.

2 scenarijus:

- Vartotojas nori pakeisti duomenis per kelis redagavimo seansus, kurie gali užtrukti kelias dienas.

Sprendimas:

- Vartotojas sukuria naują versiją pagal PRADINĘ versiją ir ją naudoja.
- Vartotojas modifikuoja elementus ir įrašo pakeitimus.
- Atlikęs pakeitimus vartotojas turi juos suderinti. Aptikęs konfliktų vartotojas gali juos išspręsti ir baigti transakciją paskelbdamas savo keitimus PRADINĖJE versijoje.
- Tada vartotojas gali panaikinti versiją.

3 scenarijus:

- Organizacija nori pradėti didelį plėtros projektą, bet jai reikia išsaugoti nepakeistą duomenų bazę ir ji turi būti pasiekiama projektavimo ir konstravimo metu.

Sprendimas:

- Sukurkite versiją pagal PRADINĘ versiją. Projektuojama naudojant šią versiją.
- Galima kurti kelias versijas (iš ankstesnių versijų), kad jose būtų matomi vadovo pakeitimai, sudarymo metu atlikti pakeitimai ir t. t.
- Galutinę versiją galima skelbti į PRADINĘ ir suderinti konfliktus.
- Kiekvieną proceso etapą galima išsaugoti kaip retrospektyvos įrašą arba baigus projektą versijas galima panaikinti.

4 scenarijus:

- Organizacija nori kontroliuosi duomenų bazės rašymo prieigą.

Sprendimas:

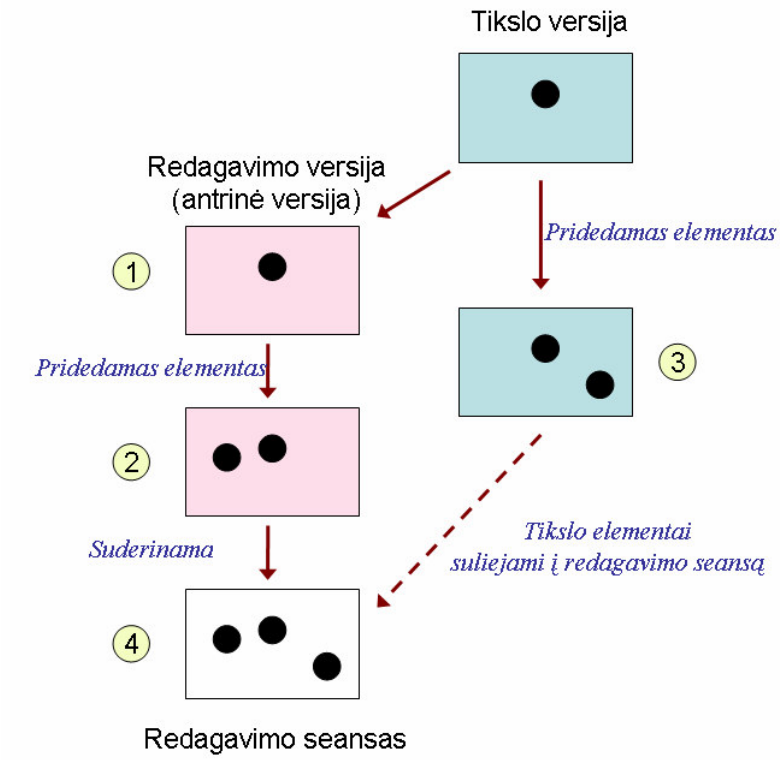
- PRADINĖ versija apsaugota nuo rašymo.
- Jeigu reikia atlikti pakeitimus, administratorius sukuria redagavimo versiją. Reikiami vartotojai turi šios versijos rašymo teises.

- Operatoriai redaguoja šią naują versiją.
- Administratorius peržiūri, redaguoja ir skelbia pakeitimus PRADINĖJE versijoje.
- Baigęs šį procesą administratorius gali panaikinti šią versiją.

3.2.3 Suderinimas

Kaip minėta, naudojant versijų sistemą SDE duomenų bazėje esantys duomenys neblokuojami, todėl yra galimybė, kad du vartotojai sukurs konfliktuojančius redagavimus. Baigus redagavimą visi vieno vartotojo versijos pakeitimai turi būti suderinti su pagrindine versija (su versija, pagal kurią sukurta redaguotoji versija; dažniausiai PRADINĖ bet ne visada). Suderinimo proceso pagrindinė versija vadinama **suderinimo tikslu** (*reconcile target*), nes joje pritaikomi nauji pakeitimai.

Suderinimą visada reikia atlikti redagavimo seanso metu. Jeigu dabartinė redagavimo versija redaguojama, vartotojai gali pradėti suderinimo procesą. Pirmas suderinimo etapas yra visų pagrindinės versijos elementų ir objektų įtraukimas į redagavimo seansą, kad vienu metu būtų galima peržiūrėti redagavimo ir tikslo versijas. Daugelis objektų abiejose versijose bus tokie patys, todėl sujungimo procesas nepadarys didelės įtakos vartotojo elementų klasės rodiniui. Tačiau pridėjus naują elementą prie tikslo versijos po redagavimo versijos sukūrimo ir pradėjus suderinimą, šis naujas elementas įtraukimas į redagavimo seansą. Geltonuose apskritimuose esantys skaičiai nurodo suderinimo veiksmų seką.



3-13 pav. Suderinimų suliejimo būsena

1 laiku buvo sukurta nauja redagavimo versija, o tikslo versija buvo jos pagrindinė versija. Prieš redagavimą redagavimo ir tikslo versijos yra vienodos. 2 laiku prie redagavimo versijos buvo pridėtas naujas elementas. Tam tikru laiku jau sukūrus redagavimo versiją, bet dar nepradėjus suderinimo, (parodyta kaip 3 laikas), kitas vartotojas prie tikslo versijos pridėjo trečią elementą. Suderinant redagavimo versiją naujasis elementas perkeliamas iš tikslo versijos į redagavimo versiją ir pateikiamas kartu su dabartine redagavimo versija. Tą galime matyti 4 laiku, kai visi trys taškai pateikiami kartu. Redagavimo seanso suderinimo metu panaikinimai ir modifikavimai bus pateikiami taip pat. Operatorius, atliekantis suderinimo operaciją, turi galimybę peržiūrėti visus sulietus redagavimo seanso pakeitimus. Jeigu objektas buvo redaguotas abiejose versijose, redaktoriui bus rodomas konfliktas.

Jeigu sukūrus redagavimo versiją tikslo versija nekeičiama, redagavimo seanso metu bus paprasčiausiai rodoma redagavimo versija. Jeigu buvo pakeista tikslo versija, o redagavimo versija liko nepakitusi, redagavimo seanso metus bus rodoma tikslo versija, nes redagavimo seanso metu bus integruoti visi tikslo versijos pakeitimai. Jeigu pakeičiamos tikslo ir redagavimo versijos, redagavimo seanso metu įtraukiami abu pakeitimų rinkiniai (ši situacija pavaizduota 3-13 pav.). Jeigu pakeičiamas tas pats objektas abiejose versijose, redagavimo seanso redaktoriui rodomas konfliktas.

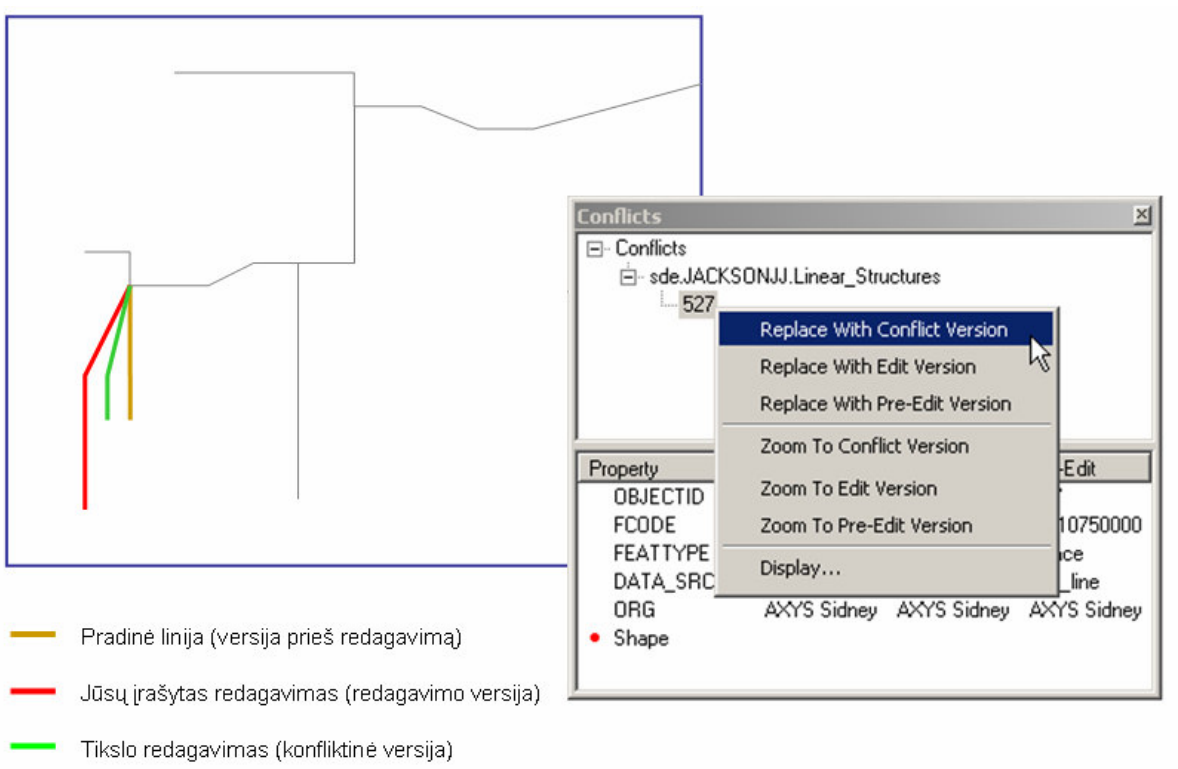
Suderinimo proceso metu vykdoma versijų sistemos optimistinio vienalaikiškumo valdymo išimtis. Suderinimo metu ir tol, kol bus atlikti veiksmai *įrašyti redagavimus (save edits)* ar *versijos skelbimas* (*version post)*, versija būna išimtinai blokuojama. Kol užblokuota, kiti vartotojai negali peržiūrėti ar redaguoti suderinamos versijos. Versijos, kurią redaguoja vienas vartotojas, negali suderinti kitas vartotojas, nes tokiu atveju negali būti suteiktas išimtinis blokavimas. Be to, suderinimo metu tikslo versija blokuojama naudojant bendrą

blokuotę. Taip užtikrinama, kad tol, kol kiti vartotojai peržiūri tikslo versiją, nebus vykdomi du suderinimai vienu metu.

3.2.4 Konfliktai ir konfliktų sprendimas

Pradiniu suderinimo suliejimo etapu tvarkomi nežymūs pakeitimai, kai pakeičiama viena versija, o kita versija lieka nepakitusi.
Pavyzdžiui, gali būti panaikinamas arba modifikuojamas vienos versijos elementas, o kitoje versijoje jis lieka nepakitęs. Tikrus konfliktus, kai objektą modifikuoja du vartotojai, reikia tvarkyti rankiniu būdu. Panašiai kaip ir sprendžiant topologines problemas, kiekvienas konfliktas gali būti patikrintas ir galima nuspręsti, kuris pakeitimas turi būti paliktas baigtoje ir suderintoje duomenų bazėje. Yra įrankių, skirtų peržiūrėti konfliktus ir pasirinkti paliekamą redagavimą.

Tarkime, kad kuriate versiją, skirtą redaguoti linijinių vamzdžių elementų klasę. Jūs redaguojate vamzdžių elementą ir šiek tiek pakeičiate jo formą. Kol atliekate pakeitimus, antrasis operatorius irgi redaguoja tą patį elementą kitaip pakeisdamas jo formą. Antrasis pakeitimas gali būti vykdomas tiesiogiai PRADINĖJE versijoje arba prieš jums patvirtinant savo pakeitimus gali būti sukuriama, redaguojama, patvirtinama ir paskelbiama kita versija. Baigus redagavimus ir juos patvirtinus, tikslo (PRADINĖS) versijos pakeitimai bus sulieti su jūsų redagavimo versija ir bus nustatytas konfliktas. Naudodami konfliktų valdymo sąsają galite peržiūrėti konfliktą ir išspręsti problemą. 3-14 pav. parodyta, kaip gali atrodyti pirmiau paminėtų konfliktuojančių vamzdžių redagavimo sąsaja.



3-14 pav. Konfliktų sprendimas

Žemėlapio rodinyje parodyti visi trys konfliktuojantys to paties vamzdyno elemento atvaizdavimai. Atvaizdavime parodytas pradinis vamzdyno brėžinys prieš jums sukuriant

redagavimo versiją (versija iki redagavimo (*the pre-edit version*)), jūsų pakeitimai (redagavimo versija (*the edit version*)) ir kito vartotojo atlikti pakeitimai (konflikto versija (*the conflict version*)). Naudodami konfliktų dialogą turite nuspręsti, kuris vamzdyno elementas yra teisingas. Jeigu nenurodysite kitaip, taikomoji programa pasirinks pirmiausiai pateiktą versiją. Tai reiškia, kad jeigu *ArcMap* nenurodysite kitaip, jūsų pakeitimas bus atmestas ir paliktas kito vartotojo redagavimas, kuris buvo pirmiau suderintas ir paskelbtas PRADINĖJE versijoje. Turite peržiūrėti kiekvieną konfliktą ir pasirinkti reikiamą kiekvieno redagavimo versiją. Tai taikoma erdviniams konfliktams ir atributų konfliktams, kai du redaktoriai skirtingai pakeičia to paties sklypo savininko vardą.

Galima pastebėti, kad redagavimų ir patvirtinimų tvarka ženkliai veikia patvirtinimo procesą. Jei tą patį elementų rinkinį pakeičia du redaktoriai, redaktorius, kuris pirmasis patvirtina ir paskelbia savo versiją nepastebės jokių konfliktų. Tas redaktorius, kuris patvirtina antrasis, turi peržiūrėti visus konfliktus ir nuspręsti, kaip teisingai turi būti atvaizduoti elementai. Tam tikrus konfliktus galima išspręsti dar redagavimo etapu. Administratoriai gali ženkliai sumažinti konfliktų tikimybę užtikrindami, kad operatoriai retai dirbtų toje pačioje geografinėje apimtyje ar naudotų tas pačias elementų klases. Be to, administratoriai gali valdyti konfliktų tvarkymą pakeisdami redagavimų skelbimų tvarką, kad labiau patyręs asmuo paskutinis patvirtintų keitimus. Taip bus teoriškai padidinama duomenų kokybė, nes konfliktus išspręs labiau patyręs darbuotojas.

Patvirtinant galimi dviejų rūšių konfliktai:

- Modifikuojamas tas pats abiejų versijų elementas. Erdvinį atvaizdavimą arba atributo vertę redaguoja du skirtingi operatoriai ir gaunami skirtingi rezultatai.
- Tas pats elementas vienoje versijoje modifikuojamas, o kitoje – panaikinamas. Vienas operatorius pakeičia elemento atributo vertę arba erdvinį atvaizdavimą, o kitas operatorius panaikina elementą. Pakeitus elemento atributą ar formą nurodoma, kad šis elementas turi likti duomenų bazėje. Vadinasi kitam vartotojui panaikinus tą elementą kyla konfliktas.

  Ši situacija skiriasi nuo atvejo kai vienas operatorius panaikina elementą, o kitas operatorius jo nepanaikina (bet ir nepakeičia). Tokioje situacijoje operatorius, kuris palieka elementą nepakeistą, nenurodo, kad šio lemento *negalima* panaikinti, todėl šis pakeitimas laikomas nereikšmingu ir konfliktas nepateikiamas. Tai gali būti laikoma patvirtinimo proceso apribojimu, bet jei operatoriai žino apie tokį veikimą, jie gali peržiūrėti panaikinimus suderinimo proceso metu.

Pirmiau aptarti konfliktai apėmė tik paprastų elementų problemas, kurie buvo keičiami juos atskyrus ir neįtakojant kitų susijusių elementų. Geoduomenų bazės elementai gali būti įvairiais būdais susiję tarpusavyje ir dėl šių ryšių vieno elemento redagavimai gali turėti įtakos kitiems tiesiogiai neredaguojamiems elementams. Elementų ryšiai, kurie gali paveikti konfliktų sprendimą:

- Su elementais susietos anotacijos
- Ryšių klasės
- Geometriniai tinklai
- Topologijos taisyklės

Su elementais susietos anotacijos

Su elementais susijusios anotacijos yra tekstinių žymių sukurtų pagal elemento atributų reikšmes tiesioginio susiejimo žemėlapyje su elemento geometrija būdas. Jeigu pakeičiama atributo vertė arba perkeliamas elementas, anotacijos tekstas atnaujinamas arba perkeliamas. Dėl to versijos suderinimas pasidaro dar sudėtingesnis. Jeigu vienas operatorius pakeičia elemento vietą (dėl to anotacija bus perkeliama į atitinkamą vietą), o kitas operatorius perkelia pačią anotaciją, bet nepakeičia elemento, kyla suderinamumo konfliktas. Tokiu atveju elemento vieta ir anotacijos vieta bus pažymėtos kaip konfliktas. Negalima suderinti elemento perkelto vienoje versijoje ir jo anotacijos perkeltos kitoje versijoje. Tam tikrais atvejais išsprendus konfliktą gali prireikti perkelti anotaciją į kitą vietą.

Ryšių klasės

Ryšių klasės sukelia panašių suderinamumo problemų kaip ir su elementais susietos anotacijos. Didžiausią rūpestį kelia pradinio elemento ryšio poveikis priskirtiems elementams, taip pat kaip elementų pakeitimo įtaka su elementais susietoms anotacijoms. Panaikinus pradinį elementą gali kilti pakopinis panaikinimas, kurio metu pašalinami tam tikri susiję elementai. Galite panaikinti ar perkelti vandens vamzdyną ir bus pašalinti arba perkelti į kitą vietą visi susiję sklendžių elementai. Konfliktas kiltų jeigu vienas operatorius panaikintų vandens vamzdyną (ir visas sklendes), o kitas operatorius pakeistų to vandens vamzdyno sklendžių atributus.

Geometriniai tinklai

Geometriniai tinklai teikia išteklių srauto tinkluose modeliavimo būdą, pvz., vandens paskirstymo tinklai, hidrologiniai tinklai (pvz. upių) ar elektros perdavimo tinklai. Geometriniuose tinkluose atvaizduojant ir modeliuojant tinklo veikimą bei valdant loginį tinklą, kuris apibūdina tinklo elementų ryšį, naudojamos briaunos, jungtys ir jungimosi taisyklės. Šio loginio tinklo jungtys gali sukelti suderinimo konfliktus.

Pavyzdžiui, jeigu vartotojas prideda šoninį vandens vamzdį (kuris atsišakoja nuo pagrindinio vamzdžio) prie vandens vamzdyno elemento, fiziškai pagrindinis vamzdis nepakeičiamas, nes ten, kur prijungiamas naujas vamzdis, jo elementas nepadalijamas į dvi dalis. Tačiau loginis tinklas, atvaizduojantis šį vandens vamzdyną, padalijamas į dvi dalis ir nurodoma naujo vamzdžio jungtis. Pasikeičia loginis vandens vamzdyno atvaizdavimas, todėl bet kuris šio pagrindinio vamzdžio elemento pakeitimas, atributas ar erdvinis pakeitimas, sukels konfliktą.

Topologija

Antroje šio kurso dalyje mes aptarėme topologijos taisykles ir jų patikrą. Kaip topologijos kūrimo dalys yra skaidymo ir jungimo į klasterius procesai, kurie užtikrina, kad arti vienas šalia kito esantys elementai bus restruktūrizuojami ir bendrai naudos tikslias vietas. Šis bendras elementų klasių, dalyvaujančių topologijoje geometrijos naudojimas reiškia, kad vieno elemento erdviniai pakeitimai gali sukelti elementų, kurie bendrai naudoja tą

geometriją, pakeitimus. Gali kilti tai pačiai elementų klasei priklausančių elementų pakeitimų arba kitai elementų klasei priklausančių elementų pakeitimų.

Kaip šių sukeltų pakeitimų valdymo būdą, *ArcMap* įrašo topologijoje dalyvaujančių elementų redagavimus ir sukuria *nepatikrintus plotus* t.y. plotus, kurių geometrija buvo pakeista, bet topologija nebuvo patvirtinta. Net patvirtinus kiekvienos versijos topologiją, po redagavimo vykdomas suderinimas gali sukelti naujų topologijos problemų. Suderinimo proceso metu atkuriami visi nesuderinti topologijos plotai, kad būtų galima iš naujo patvirtinti šių plotų topologiją ir užtikrinti, kad nekilo naujų dviejų versijų topologinių klaidų. Jeigu redaguojamos elementų klasės, dalyvaujančios topologijoje, prieš skelbiant pakeitimus tikslo versijoje, visada turi būti atliekamas topologijos patvirtinimas, kaip suderinimo proceso dalis.

Paskelbimas

Peržiūrėjus visu konfliktus ir juos išsprendus galima paskelbti versiją tikslo versijoje. Paskelbus versiją, visi nežymūs redagavimai (nekonfliktuojantys redagavimai) ir visi konfliktų sprendimo metu atlikti redagavimai bus visam laikui įrašyti tikslo versijoje (dažnai ji vadinama PRADINE versija). Paskelbę pakeitimus vartotojai gali panaikinti redagavimo versiją arba išsaugoti ją kaip retrospektyvos nuorodą.

3.2.5 Kelių vartotojų redagavimas netaikant versijų sistemos

9.2 *ArcMap* versija palaiko kelių vartotojų redagavimą nenaudojant versijų sistemos vienalaikiškumo valdymo. Kaip minėta, taikant versijų sistemą nesukuriama atskira versijos kopija, bet dirbama su pagrindine versija, o po to toje pagrindinėje versijoje išsaugomi atlikti keitimai. Todėl niekada neatvaizduojama tikroji duomenų bazės versija. Jeigu 3-iosios šalies taikomoji programa, kuri nebuvo sukurta sąveikauti su duomenų bazių būsenomis ir delta lentelėmis, norėtų gauti prieigą prie versijos duomenų, ji niekada nepamatytų tikrosios versijos būsenos. Versijos gali padidinti nereikalingo darbo kiekį, nors gali reikėti tik nedidelių ir paprastų duomenų pakeitimų. Jeigu elementų klasės nedalyvauja topologijos ar geometrijos tinkluose, galima naudoti trumpų transakcijų modelį, kuris naudoja DBVS blokavimo mechanizmą.

3.3 Geoduomenų bazės tvarkymas

GIS duomenų tvarkymui reikia skirti nemažai dėmesio. Tam tikrų duomenų formatų, pvz., *COVERAGE* sluoksnių ar SHAPE failų, tvarkymas yra pakankamai nesudėtingas, bet reikia nepamiršti atsarginių kopijų kūrimo ir metaduomenų atnaujinimo. Geoduomenų bazių duomenų tvarkymas gali būti ženkliai sudėtingesnis.

Atsarginių duomenų kūrimo ir metaduomenų tvarkymo funkcijos yra universalios, todėl mes jų neaptarsime, tik paminėsime jų būtinybę. Šioje temoje nagrinėsime užduotis, kurias būtina atlikti prižiūrint geoduomenų bazės struktūrą. Kuo daugiau vartotojų ir kuo didesnė bei sudėtingesnė yra jūsų geoduomenų bazė, tuo daugiau tvarkymo ji pareikalaus. Kaip ir bet kurioms kitoms organizacijos funkcijoms, pasiekus tam tikrą apimtį, tvarkymo užduotims reikia skirti daug dėmesio.

3-1 lentelėje pateikta skirtingų užduočių, kurias turi atlikti geoduomenų bazės administratorius, suvestinė. Lentelės viršuje išvardyti skirtingi geoduomenų bazės tipai. Geoduomenų bazės dydis ir sudėtingumas didėja iš kairės į dešinę. Nesunku pastebėti, kad net užduočių, kurias reikia atlikti tvarkant įmonei skirtą geoduomenų bazę, skaičius yra ženkliai didesnis negu asmenims skirtos geoduomenų bazės tvarkymo užduočių skaičius. Be to, įmonėms skirtų geoduomenų bazių tvarkymo užduotys yra daug sudėtingesnės negu paprastesnių geoduomenų bazių tvarkymo užduotys.

3-1 lentelė. Geoduomenų bazės tipų tvarkymo užduočių suvestinė

Užduotis	Asmenims skirta / failų geoduomenų bazė	Asmenims skirta SDE	Darbo grupėms skirta SDE	Įmonėms skirta SDE
Diegimas	N	T	T	T
Konfigūravimas	N	N	N	T
Glaudinimas	T (elementų klasės glaudinimas)	T (versijų glaudinimas)	T (versijų glaudinimas)	T (versijų glaudinimas)
Sumažinimas	T	T	T	T
Vartotojų abonementai ir teisės	N	T (iki 3 vartotojų)	T	T
Atsarginės kopijos ir atkūrimas	T	T	T	T
Indeksų performavimas	N	T	T	T
Statistikos performavimas	N	T	T	T
Derinimas	N	N	N	T

Tolesniuose skyriuose išsamiai apibūdinamos šioje lentelėje pateiktos užduotys.

3.3.1 Diegimas ir konfigūravimas

Šios užduotys susijusios su darbu, kurį reikia atlikti prieš pradedant naudoti geoduomenų bazę. Šias užduotis sudaro papildomos programinės įrangos diegimas (ne *ArcGIS* darbalaukio taikomųjų programų) ir papildomų *ArcSDE* reikiamų parametrų nustatymas. Dažniausiai šias užduotis reikia atlikti tik įdiegus naują programinę įrangą.

Asmenims skirtų ir failų geoduomenų bazių nereikia įdiegti ar konfigūruoti. Diegiant asmenims ir darbo grupėms skirtas SDE geoduomenų bazes reikia įdiegti *SQL Server Express*, bet daugiau konfigūruoti nereikia.

Įmonėms skirtų SDE diegimas yra sudėtingesnis ir apima tris veiksmus. Šie veiksmai yra skirtingi atsižvelgiant į tai, pagal kurią RDBVS jūsų įmonėms skirta SDE bus kuriama. Tačiau pagrindiniai veiksmai yra panašūs. Visų pirma reikia įdiegti RDBVS (pvz., DB2), nustatyti aplinkos kintamuosius ir tam tikrais atvejais sukurti naują ir tuščią duomenų bazę. Antru veiksmu reikia įdiegti *ArcSDE* atsižvelgiant į DBVS ir operacinės sistemos diegimo vadovo informaciją. Trečiu veiksmu sukuriama geoduomenų bazės schema, paruošiamas SDE administratoriaus abonementas ir jam suteikiamos reikiamos teisės bei paleidžiama SDE paslauga (*service*). Paslauga yra procesas, kuris „išklauso“ vartotojų, pageidaujančių ryšio su geoduomenų baze.

Kaip konfigūravimo proceso dalis bus nustatyti įvairūs veikimo parametrai, į kuriuos atsižvelgiant bus nustatoma lentelių vieta, laikinosios vietos dydis ir duomenų saugojimo bei prieigos prie jų būdai. Pavyzdžiui, galima valdyti tam tikrų DBVS erdvinių objektų fizinio saugojimą. Visose sistemose galite naudoti suglaudintus dvejetainius formatus (pvz., didelius dvejetainius objektus, BLOB duomenų tipą), o kai kuriose sistemose galite naudoti duomenų tipus, kurie buvo sukurti saugoti erdvinius duomenis (pvz., *ST_SPATIAL* duomenų tipai iš *Oracle Spatial* arba *Informix Spatial DataBlade*). ESRI teikia kiekvienos RDBVS diegimo ir konfigūravimo dokumentus. Šie procesai skiriasi atsižvelgiant į diegiamą taikomąją programą. Apskritai, norint pradėti naudoti SDE pakanka numatytųjų konfigūravimo parametrų, bet geriau supratę duomenų bazės naudojimo ir dydžio sąvokas galėsite pagerinti efektyvumą pakeisdami tam tikras konfigūravimo parinktis. Tokį modifikavimą aptarsime paskesniame derinimo skyriuje.

3.3.2 Vartotojų abonementai

Geoduomenų bazės vartotojų vardų ir slaptažodžių sukūrimas suteikia galimybę valdyti geoduomenų bazės duomenų rinkinių vartotojų prieigos tipą. Geoduomenų bazės vartotojo nustatymo procesas prieš suteikiant jam prieigą prie duomenų vadinamas **autentifikavimu**. Geoduomenų bazių vartotojus galima autentifikuoti dviem būdais – naudojant operacinės sistemos autentifikavimą arba duomenų bazės autentifikavimą.

Naudojant operacinės sistemos (OS) autentifikavimą, vartotojai prisijungia prie kompiuterio naudodami savo kredencialus (vartotojo vardą ir slaptažodį). Naudojant geoduomenų bazę OS kredencialai perduodami į duomenų bazę ir vartotojams nereikia autentifikuotis antrą kartą. Tai reiškia, kad administratoriai turi pridėti prie duomenų bazės esamus OS vartotojų vardus ir sukonfigūruoti duomenų bazę, kad ši priimtų OS autentifikavimą.

Naudojant duomenų bazės autentifikavimą vartotojai prisijungia prie kompiuterio įprastai, bet norėdami naudoti geoduomenų bazės duomenis, turi atskirai prie jos prisijungti. *Oracle* ir *SQL Server* programose šiuos abonementus reikia sukurti ir prižiūrėti naudojant RDBVS.

Paruošus abonementus galima nustatyti vartotojų teises. Kaip jau buvo aptarta pirmesnėje temoje, kiekvienam vartotojui galima suteikti skirtingas prieigos prie kiekvienos elementų klasės ar kitų duomenų bazės objektų teises. Pavyzdžiui, vienam vartotojui gali būti suteikta teisė tik skaityti gatvių duomenis, bet visos rašymo teisės naudojant žemės sklypus. Kitas vartotojas gali net nežinoti apie žemės sklypų buvimą, bet turėti visas prieigos prie gatvių duomenų teises. Naudodami šį mechanizmą administratoriai gali kontroliuoti, kuris iš daugelio vartotojų, turinčių prieigą prie duomenų, gali tvarkyti tam tikrą duomenų rinkinį ir kuris bus tų duomenų vartotojas.

Be to, administratoriai gali sukurti vartotojų **grupes**, kad būtų lengviau suteikti teises. Užuot kiekvienam vartotojui suteikus kiekvienos elementų klasės ar lentelės teises skaityti ar rašyti, galima sukurti grupės teisių rinkinį ir tas pačias teises taikyti keliems vartotojams. Galima nustatyti dvi grupes, pvz., „Tyrėjas“ ir „Redaktorius“. Tyrėjui gali būti suteikta kelių duomenų bazės objektų teisė tik skaityti, o redaktoriui gali būti suteikta visų objektų teisė rašyti. Nustačius šias grupes ir jų kiekvieno duomenų bazės objektų rinkinio teises, į šias grupes galima paprasčiausiai įtraukti arba iš jų pašalinti naujus vartotojus ir taip suteikti jiems reikiamas teises.

Asmenims skirtų ir failų geoduomenų bazių vartotojų abonementų tvarkyti nereikia, nes jos nepalaiko vienu metu vykdomų transakcijų. Kaip aptarta 2 šio kurso dalyje, asmenims skirtose geoduomenų bazėse funkciškai taikomas duomenų bazės blokavimas, nes geoduomenų bazę gali redaguoti daugiausiai vienas vartotojas. Naudojant failų geoduomenų bazes leidžiamas vienas geoduomenų bazės elementų duomenų rinkinio redaktorius. Asmenims skirtų ir failų geoduomenų bazės nepalaiko atskirų elementų klasių teisių nustatymo. Vartotojai neautentifikuojami.

Asmenims skirtas SDE geoduomenų bazes tvarkyti nėra sudėtinga, nes vienu metu geoduomenų bazę gali redaguoti tik keli vartotojai. Šis SDE duomenų bazės tipas tiks tik labai nedidelėms organizacijoms ar projektuotojų komandoms, todėl tikėtina, kad bus nesudėtinga nustatyti vartotojų abonementus ir teises. Darbo grupėms ir įmonėms skirtų SDE diegimas reikalaus kruopščios priežiūros, nes šias sistemas gali naudoti labai didelės organizacijos, turinčios daug vartotojų, kuriems suteiktos skirtingų tipų prieigos teisės.

3.3.3 Glaudinimas

Failų geoduomenų bazėje saugomas vektorių elementų klases ir ne erdvinių duomenų lenteles galima glaudinti (*compressed*). Tai pasirinktinis procesas, kurio metu sumažinamas šių objektų dydis. Suglaudinti duomenys skirti tik skaityti. Jie įprastai rodomi *ArcCatalog* ar *ArcMap* ir išlaiko beveik tokį patį rodymo bei užklausų efektyvumą, bet jų negalima redaguoti. Atsižvelgiant į glaudinamų duomenų tipą, suglaudinti duomenys gali užimti tik šiek tiek mažiau vietos arba jų glaudinimo santykis gali būti 4:1. Glaudinimas yra pasirinktinė tvarkymo užduotis. Norėdami leisti redaguoti duomenis elementų klasių galite neglaudinti.

SDE duomenų bazėje glaudinimas atlieka kitokią funkciją. Naudojant bet kurio tipo *ArcSDE* geoduomenų bazę (skirtą asmenims, darbo grupėms ar įmonėms), glaudinant bus reorganizuoti geoduomenų bazės, kuriai taikoma versijų sistema, duomenys. Nepamiršite, kad vartotojams pakeitus SDE geoduomenų bazę naudojant versijų sistemą, patys duomenų bazės objektai nėra pakeičiami. Visi kiekvienos versijos pakeitimai tampa duomenų bazės būsenomis ir yra saugomi delta lentelėse. Jeigu atliekama daug redagavimų, delta lentelių būsenų ir eilučių skaičius gali taip padidėti, kad sumažės duomenų bazės efektyvumas.

Glaudinant SDE bet kuris būsenų rinkinys sutraukiamas į vieną būseną, kuri neturės įtakos atvaizduojant versiją. Jei įmanoma delta lentelių eilutės perkeliamos į bazines lenteles. Didžiausias glaudinimo operacijų pranašumas yra tas, kad visos versijos yra suderinamos ir paskelbiamos, o pačios versijos panaikinamos. Duomenys iš delta lentelių gali būti visiškai panaikinami ir lieka tik viena būsena (PRADINĖ būsena). Praktiškai aktyviomis gali likti kelios versijos, bet glaudinimo operacija vis tiek padidina duomenų bazės efektyvumą.

Darbo aplinkose, kuriose reguliariai atliekama daug duomenų redagavimų, glaudinti SDE galima kiekvieną dieną. Pagal ESRI rekomendacijas net ir atliekant nedaug redagavimų glaudinti reikėtų kartą per savaitę.

3.3.4 Sumažinimas

Nors sumažinimas (*compaction*) panašus į glaudinimą, tvarkymo požiūriu tai skirtingas procesas. Pridedant duomenis geoduomenų bazėje ir juos panaikinant kinta duomenų elementų tvarka ir failų sistemoje atsiranda tuščių vietų. Kuo ilgiau taip vyksta, tuo lėčiau nuskaitomi duomenys ir tuo daugiau vietos užima geoduomenų bazė. Sumažinant pertvarkomi geoduomenų bazės duomenys, duomenų objektai išdėstomi aiškesne tvarka ir pašalinamos nuorodos į nenaudojamas vietas, panašiai kaip defragmentuojant standųjį diską, kurį galite atlikti naudodami kompiuterio operacinę sistemą. Dažnai modifikuojamas duomenų bazes reikia sumažinti katą per mėnesį.

Asmenims skirtose ir failų geoduomenų bazėse šis procesas vadinamas sumažinimu. Asmenims ir darbo grupėms skirtose geoduomenų bazėse ši funkcija vadinama duomenų bazės **sutraukimu** (*shrinking*). Įmonėms skirtose geoduomenų bazėse šią funkciją valdo DBVS.

3.3.5 Duomenų bazės derinimas

Duomenų bazės derinimas (*tuning*) yra sistemos efektyvumo padidinimo gerinant sistemos išteklių naudojimą procesas. Kuo daugiau vartotojų turi tam tikra geoduomenų bazė, tuo daugiau derinimo turi atlikti administratorius. Efektyviai suderinus duomenų bazę ženkliai padidės jos efektyvumas. Administratorius gali tik labai nežymiai padidinti asmenims skirtų ir failų geoduomenų bazių, kaip ir asmenims ir darbo grupėms skirtų geoduomenų bazių efektyvumą. Yra efektyvių tvarkymo užduočių (pvz., glaudinimas), bet šios funkcijos yra priežiūros funkcijos, o ne derinimo. Įmonėms skirta geoduomenų bazė yra visas funkcijas turinti RDBVS, tačiau yra nemaža būdų pakeisti DBVS veikimą.

Atsižvelgiant į skirtingas DBVS, derinimo būdai ir užduotys gali būti labai skirtingos. Toliau pateikiamos bendrosios derinimo užduotys:

Įvestis / išvestis	Kai vartotojai bando gauti prieigą prie duomenų failų, esančių toje pačioje fizinio disko vietoje, šie vartotojai varžosi dėl tų pačių išteklių. Tai vadinama Įv./Išv. nesutarimais. Galime sumažinti nesutarimų skaičių išsaugodami dažnai naudojamus failus skirtingose disko vietose.
Atmintis	Daugelis DBVS funkcijų naudoja atmintį, įskaitant virtualiosios atminties naudojimą, duomenų buferizavimą ir laikinosios vietos naudojimą. Naudojant numatytąsias DBVS vertes šie atminties vienetai gali būti neefektyvūs.
Indeksai	Erdvinių ir ne erdvinių duomenų indeksai gali ženkliai padidinti duomenų nuskaitymo efektyvumą. Erdvinių duomenų indeksus aptarsime kitoje šio kurso dalyje, bet iš esmės tai yra greito elementų radimo būdas, atsižvelgiant į jų vietą, panašiai kaip jūs galite greitai rasti knygos temas peržiūrėję turinio rodyklę ir radę reikiamo puslapio numerį. Erdvinių duomenų indekso nustatymo būdas gali paveikti geografinių užklausų greitį ir apdorojimą.
Rodiniai	Rodinys yra išsaugota užklausa, kuri gali nuskaityti kelių lentelių ar objektų informaciją bei suderinti taip, kad būtų galima vėliau panaudoti

šią informaciją. Jeigu vartotojai dažnai kuria daug užklausų, administratorius gali paruošti duomenų bazės rodinių.

Derinti gali būti sudėtinga, todėl administratoriai turėtų ieškoti informacijos apie naudojamos RDBVS derinimą SDE derinimo vadove.

3.3.6 Statistikos ir indeksų atnaujinimas

Taikomosios duomenų bazių programos užtikrindamos greitą užklausų vykdymą naudoja daug resursų. Vienas iš dalykų į kurį atsižvelgia DBVS nuspręsdama, kaip vykdyti tam tikrą užklausą yra statistika. Statistiką sudaro įvairi duomenų bazės informacija, pvz., tam tikros lentelės įrašų skaičius, kiekvienos lentelės duomenų puslapių skaičius, kiekvienos lentelės fizinė vieta diske ir tai, ar yra sudarytų lentelės indeksų. Atsižvelgdama į šią statistiką DBVS gali optimizuoti vidinę tam tikros užklausos vykdymo strategiją. Jei ši statistika neatnaujinama, užklausų optimizavimo priemonė negali efektyviai veikti.

Duomenų bazės statistiką reikėtų atnaujinti prieš glaudinimo operaciją ir ją užbaigus, pridėjus ar pašalinus topologijos taisyklių ir importavus, įkėlus ar nukopijavus duomenis į *ArcSDE* geoduomenų bazę. Reikia atnaujinti visų SDE duomenų bazių statistiką, bet to negalima atlikti naudojant asmenims skirtas ar failų geoduomenų bazes.

Duomenų pridėjimas, panaikinimas ir duomenų bazės glaudinimas gali fragmenuoti indeksus, kuriuos DBVS naudoja greitai rasti įrašus. Nors efektyvumo nuostoliai dėl fragmentuotų indeksų nėra dideli, administratoriai gali reguliariai juos atnaujinti. Asmenims ir darbo grupėms skirtų SDE duomenų bazių indeksus galima atnaujinti naudojant *ArcCatalog*. Informacijos apie įmonėms skirtų SDE indeksų atnaujinimą vartotojai turėtų ieškoti konkrečios DBVS dokumentacijoje.

Dalies klausimai savarankiškam darbui:

1. Kodėl RDBVS nesuteikia nekontroliuojamos kelių vartotojų prieigos prie duomenų bazės? Kokių problemų gali kilti?
2. Kokie esamų pramonės standartus atitinkančių DBVS (pvz., *Oracle*, SQL *Server*) taikymo saugant GIS duomenis pranašumai?
3. Apibūdinkite kas yra versijų sistema ir kam ji skirta.
4. Kodėl versijų sistema vadinama optimistiniu vienu metu vykdomų transakcijų valdymo būdu?
5. Jeigu būtumėte darbo grupėms skirtos *ArcSDE* geoduomenų bazės administratorius, kokias tvarkymo užduotis reikėtų atlikti ir kaip dažnai tą daryti? Įsivaizduokite, kad prižiūrite nedidelę vartotojų bendruomenę, kuri atlieka nedaug duomenų bazės redagavimų.

Literatūra

- Date, C.J. *An Introduction to Database Systems*. Addison-Wesley, 2000.
- Elmasri, R., and Navathe, S.B. *Fundamentals of Database Systems*. Addison-Wesley, 2000.
- ESRI. *Building a Geodatabase*. ESRI Press, 2005.
- ESRI. *Versioning*. ESRI Technical Paper, 2004. Viewed July, 2007.
- http://downloads.esri.com/support/whitepapers/ao_/Versioning_2.pdf.
- ESRI. *Understanding SDE*. ESRI Press, 2005.
- ESRI. *ArcGIS On-line Help*. Viewed July, 2007.
- http://webhelp.esri.com/arcgisdesktop/9.2/index.cfm
- Kroenke, D.M. *Database Processing Fundamentals, Design and Implementation*. Prentice-Hall, 1998.
- Zeiler, M. *Modeling Our World*. Redlands, CA: ESRI Press, 1999.

4 Papildomos temos

Ankstesniuose dalyse buvo einama nuo teorijos link taikomosios srities, pradedant nuo pagrindinės duomenų bazės teorijos ir baigiant itin specifinėmis temomis, susijusiomis su konkrečių programų ir duomenų bazių administravimu. Ši dalis šiek tiek skiriasi nuo minėto požiūrio, nes joje pateikiamos temos, susijusios su geografinėmis duomenų bazėmis, kurioms tirti būtinas tam tikras žinių lygis.

Dalis pradedama nuo struktūrizuotų užklausų kalbos (SQL) aptarimo. Ši kalba – pagrindinis bet kokios duomenų bazės naudojimo įrankis. 2 temoje pateikiama prieigos prie didelių geografinių duomenų bazių metodų apžvalga. 3 temoje tiriama laiko įtraukimo į mūsų geografines duomenų bazes sąvoka, įskaitant kelis šios problemos sprendimo pavyzdžius. 4 temoje aptariamos kelios standartinio vieno serverio ir kelių klientų požiūrio į geoduomenų bazės konfigūracijas alternatyvos.

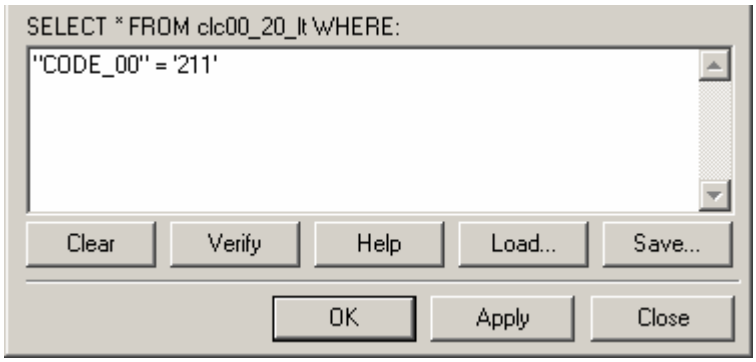
Dalies turinys

- Struktūrizuotų užklausų kalba (*SQL*)
- Erdvinis indeksavimas
- Laiko duomenų saugojimas
- Paskirstytosios DBVS

4.1 Struktūrizuotų užklausų kalba (SQL)

Struktūrizuotų užklausų kalba (SQL) – tai standartinė reliacinio modelio kalba. Ji vartotojams leidžia kurti duomenų bazių struktūras, manipuliuoti duomenimis (pvz., naujinti) ir teikti užklausas duomenims iš lentelių išrinkti. Beveik visos SDBVS programos palaiko SQL, o kai kuriais atvejais jos papildo standartines funkcijas nuosavais kalbos plėtiniais. SQL XX amžiaus VIII dešimtmetyje sukūrė IBM. Ji kaip manipuliavimo duomenimis kalba buvo skirta jų reliacinių duomenų bazių valdymo sistemai „System R“. Nuo to momento šis išradimas priimtas Amerikos Nacionalinio Standartų Instituto (ANSI) ir Tarptautinės Standartizacijos Organizacijos (ISO).

GIS programų vartotojai ir programuotojai SQL naudoja itin plačiai. Viena iš elementariausių procedūrų *ArcMap* aplinkoje – dialogo langas *Parinkimas pagal atributus (Select by Attributes)*. Juo SQL naudojama išrinkti įrašus iš geoobjektų klasės. 4-1 matote šio dialogo lango dalį. Programuotojai, naudojantys *ArcObjects* naujoms GIS programoms arba esamų *ArcMap* funkcijų plėtiniams kurti dažnai naudoja SQL manipuliuoti geoduomenų bazėje laikomais duomenimis.



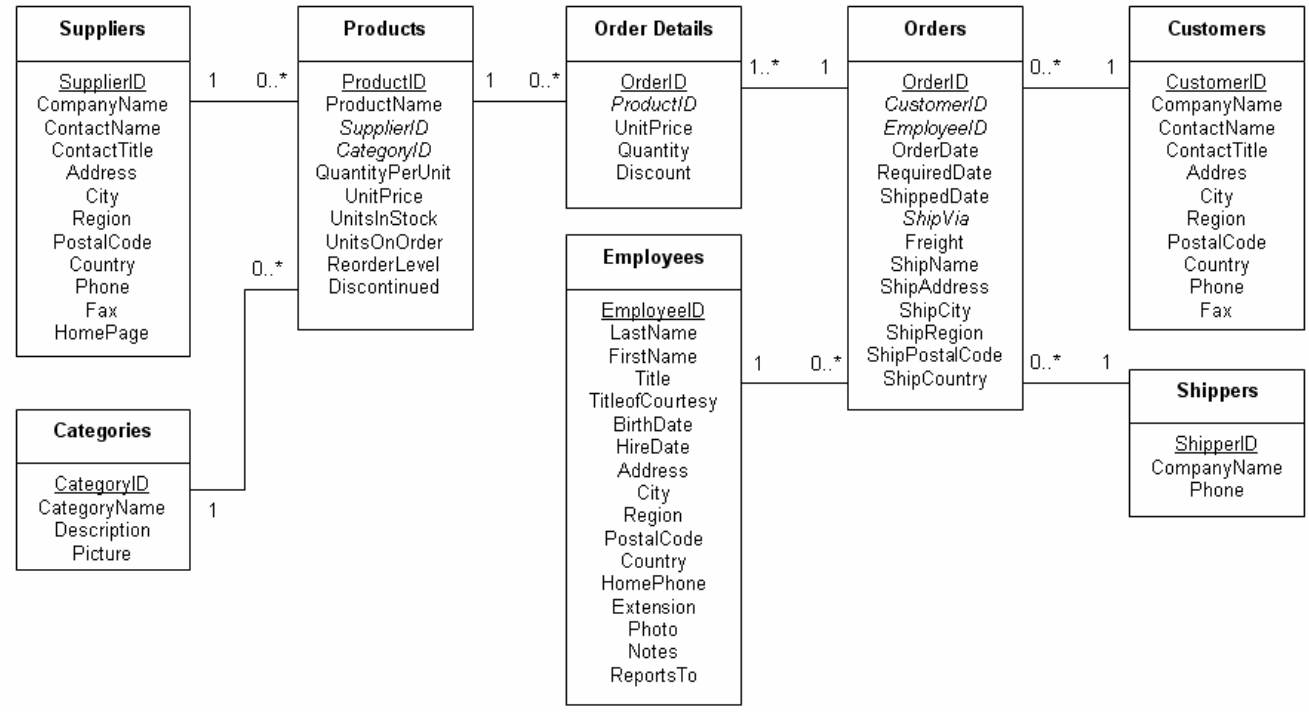
4-1 pav. „ArcMap“ dialogo langas Parinkimas pagal atributus (Select by Attributes)

SQL pasižymi keliomis charakteristikomis:

- leidžia kurti duomenų bazių ir lentelių struktūras, naudojant duomenų apibrėžimo (**Data Definition**) komandas;
- leidžia atlikti įtraukimo, šalinimo ir keitimo funkcijas, naudojant manipuliavimo duomenimis (**Data Manipulation)** komandas;
- leidžia konstruoti sudėtingas užklausas neapdorotiems duomenims transformuoti į naudingą informaciją, taip pat – naudojantis manipuliavimo duomenimis (*Data manipulation*) kalba;
- nesunku naudoti ir išmokti; SQL susideda iš maždaug 100 žodžių žodyno;
- kalba yra mobili, tad tos pačios komandos veikia visose SDBVS. Šis teiginys teisingas dažniausiai – egzistuoja tam tikros atskirų SQL įdiegčių variacijos, **SQL dialektai**. Pavyzdžiui, kai kurios komandos, veikiančios *Oracle* aplinkoje, turi būti šiek tiek pakoreguotos, kad galėtų veikti *SQL Server* aplinkoje.

4.1.1 Duomenų bazės situacijos nagrinėjimas

Visuose toliau pateikiamuose pavyzdžiuose naudojama duomenų bazė atitinka *Microsoft Northwind* duomenų bazę. Ji išreiškia paprastą fiktyvios pardavimų organizacijos duomenų bazės struktūrą. Šioje duomenų bazėje yra tokie objektai kaip *Užsakymai (Orders)*, *Klientai (Customers)* ir *Darbuotojai* (*Employees)*. 4-2 pav. pateikiama *Northwind* duomenų bazės klasių diagrama.



4-2 pav. *Microsoft Northwind* duomenų bazė

Užbaigę šio kurso 1 dalį, studentai turėtų būti susipažinę su čia naudojimu žymėjimu. Jis teigia, kad, pavyzdžiui, Klientas gali turėti nulį arba daugiau užsakymų, tačiau Užsakymas turi būti susijęs su vienu ir tik su vienu Klientu. Į šią diagramą pagal pirmiau naudotą žymėjimą įtraukti pirminiai ir išoriniai raktai: pirminiai raktai pabraukti, o išoriniai raktai išskirti *kursyvu*. *OrderID* – tai pirminis *Užsakymų* (*Orders*) lentelės raktas, o *Užsakymų* lentelėje esantis atributas *CustomerID* – tai išorinis raktas, leidžiantis su kiekvienu užsakymu susieti tinkamą klientą.

Nors šioje duomenų bazėje nėra geografinių objektų ir dėl šios priežasties ji nebus naudinga aptariant erdvines užklausas, ją pravartu nagrinėti mokantis SQL pagrindų dėl kelių priežasčių:

- tai – duomenų bazė, naudojama mokantis *Microsoft* duomenų bazių produktų, pvz., *Access* ir *SQL Server*. Ši duomenų bazė gali būti įdiegiama kaip duomenų bazių programos dalis, tad daugelis studentų su šiuo pavyzdžiu bus susipažinę anksčiau;
- ši pavyzdinė duomenų bazė naudojama jau bent 10 metų, tad egzistuoja daug spausdintos bei elektroninės (internete) *Northwind* mokymo bei analogiškos medžiagos;
- tai – gerai apsvarstyta duomenų bazė, apimanti daug atributų duomenų tipų, asociacijų tipų ir pavadinimų formalumų;

- ji buvo parengta naudojant gerą aiškių bei suprantamų duomenų kolekciją, tad užklausų rezultatai bus pagrįsti.

Teorinės medžiagos pateikimo metu mes greičiausiai išanalizuosime tik nedidelę šios duomenų bazės dalį, tačiau ją naudosime ir praktiniuose pratimuose.

Egzistuoja dviejų tipų SQL komandos: duomenų apibrėžimo komandos, leidžiančios vartotojams apibrėžti duomenų struktūrą, ir manipuliavimo duomenimis kalbos komandos, leidžiančios vartotojams įterpti, šalinti ir naujinti bei išrinkti duomenis.

4.1.2 Duomenų apibrėžimo komandos

Duomenų apibrėžimo komandos leidžia vartotojams kurti duomenų bazes ir lenteles, jos dažnai vadinamos *Duomenų apibrėžimo kalbos* (*Data Definition Language*, DDL) komandomis. Tai ta SQL dalis, kuri lyginant atskiras įdiegtis skiriasi labiausiai, paprasčiausiai dėl to, kad lentelių kūrimas apima atributų duomenų tipus ir jie dažnai atskirose SDBVS šiek tiek skiriasi. Toliau pateikiamuose pavyzdžiuose panaudosime *SQL Server* duomenų tipus, tad toliau išdėstytos pavyzdinės komandos kitose duomenų bazių sistemose, pvz., *Oracle*, gali neveikti tinkamai. Tačiau skirtumai yra nežymūs ir vartotojai turėtų pajėgti nesunkiai adaptuoti čia pateikiamą medžiagą kitoms SDBVS programoms. Žr. savo SDBVS SQL dokumentaciją, kur rasite išsamią informaciją apie savąjį SQL dialektą.

Pati paprasčiausia SQL komanda yra CREATE DATABASE (sukurti duomenų bazę). Šia komanda sukuriama tuščia duomenų bazė, kurioje galima kurti lenteles. Sintaksė yra tokia:

```
CREATE DATABASE <duomenų bazės pavadinimas>

pvz., CREATE DATABASE ManoNaujaDuombaze
```

Kita pagrindinė duomenų apibrėžimo komanda – tai CREATE TABLE (sukurti lentelę). Ja sukuriama kiekviena lentelė, įskaitant visus atributus ir jų duomenų tipus, taip pat galima apibrėžti pirminius ir išorinius raktus. Bazinė sintaksė atrodo taip:

```
CREATE TABLE <lentelės pavadinimas> (
    <1atributo pavadinimas ir charakteristikos>,
    <2atributo pavadinimas ir charakteristikos>,
    <pirminio rakto apibrėžtis>,
    <išorės raktų apibrėžtys>)
```

Pavyzdžiui, norint sukurti *Klientų (Customers)* lentelę iš 4-2 pav., komanda turėtų atrodyti taip:

```
CREATE TABLE Customers (
    CustomerID          nchar (5)           NOT NULL,
    CompanyName         nvarchar(40)        NOT NULL,
    ContactName         nvarchar(30)        NULL,
    ContactTitle        nvarchar(30)        NULL,
    Address             nvarchar(60)        NULL,
```

```
    City            nvarchar(15)            NULL,
    Region          nvarchar(15)            NULL,
    PostalCode      nvarchar(10)            NULL,
    Country         nvarchar(15)            NULL,
    Phone           nvarchar(24)            NULL,
    Fax             nvarchar(24)            NULL,
    CONSTRAINT PK_Customers PRIMARY KEY (CustomerID)
```

Visa tai galima surašyti vienoje eilutėje, tačiau skaitomumo tikslais programuotojai vienoje eilutėje įrašo vieną atributą. Be to, kalbos raktažodžiai, pvz., CREATE, čia rašomi didžiosiomis raidėmis – taip taipogi stengiamasi pagerinti skaitomumą. Šiuos žymėjimus naudosime visame dokumente.

Lentelėje naudojami tik du duomenų tipai:

nchar	Fiksuoto ilgio simbolių laukas, kurio ilgis siekia iki 255 simbolių. Jei į jį įrašote teksto eilutes, kurių ilgis nesiekia nurodytos ribos, likę simboliai lieka tušti.
nvarchar	Kintamo ilgio simbolių laukas.

Kaip buvo minėta, šiuose pavyzdžiuose naudojami duomenų tipai yra suderinami su *SQL Server* ir čia pateikiamas kodas gali neveikti tinkamai, pvz., *Oracle* aplinkoje. *Oracle* sistemoje *nvarchar* būtų keičiamas *varchar* arba *varchar2*.

Specifikacija NOT NULL (ne nulis) – tai priemonė priversti įvesti konkretaus atributo vertę. Specifikacija NOT NULL reiškia, kad vartotojas negali įvesti naujo įrašo ir palikti šį atributą tuščią. Specifikacija NULL (nulis) reiškia, kad atributas yra papildomas ir, esant reikalui, gali būti praleidžiamas.

Kai kurios SDBVS, pvz., *MS Access*, nepalaiko PRIMARY KEY (pirminio rakto) apibrėžimo. Tokiais atvejais paprasčiausiai praleiskite šią komandos sąlygą.

Jei lentelėje reikia turėti sudėtinį pirminį raktą, tai pažymima atskirais kableliais atskirtais narių atributais, pvz., *PRIMARY KEY (OrderID, CustomerID)*.

Kadangi *CustomerID* pateikiamas *Užsakymų* (*Orders*) lentelėje kaip išorinis raktas, sukursime *Klientų* (*Customers*) lentelę prieš *Užsakymų* (*Orders*) lentelę. *Užsakymų* (*Orders*) lentelės DDL iliustruoja išorinių raktų apribojimų naudojimą, ji atrodo taip:

```
CREATE TABLE Orders (
    OrderID         int             NOT NULL,
    CustomerID      nchar (5)       NULL,
    EmployeeID      int             NULL,
    OrderDate       datetime        NULL,
    RequiredDate    datetime        NULL,
    ShippedDate     datetime        NULL,
    ShipVia         int             NULL,
    Freight         money           NULL,
    ShipName        nvarchar (40)   NULL,
```

```
        ShipAddress         nvarchar (60)       NULL,
        ShipCity            nvarchar (15)       NULL,
        ShipRegion          nvarchar (15)       NULL,
        ShipPostalCode      nvarchar (10)       NULL,
        ShipCountry         nvarchar (15)       NULL,
        CONSTRAINT PK_Orders PRIMARY KEY (OrderID),
        CONSTRAINT FK_Orders_Customers FOREIGN KEY (CustomerID)
                REFERENCES Customers (CustomerID),
        CONSTRAINT FK_Orders_Employees FOREIGN KEY (EmployeeID)
                REFERENCES Employees (EmployeeID),
        CONSTRAINT FK_Orders_Shippers FOREIGN KEY (ShipVia)
                        REFERENCES Shippers (ShipperID)
)
```

Šioje lentelėje naudojami keli papildomi duomenų tipai:

Int	Sveikieji skaičiai nuo -2^{31} iki $+2^{31} - 1$
Datetime	Datos ir laiko duomenys
Money	Monetarinės vertės nuo -2^{63} iki $+2^{63} - 1$

Be to, kai kuriose duomenų bazių įdiegtyse galima naudoti sąlygas ON DELETE arba ON UPDATE ir papildomai valdyti išorinių raktų suvaržymus.

```
CONSTRAINT FK_Orders_Customers FOREIGN KEY (CustomerID)
        REFERENCES Customers (CustomerID)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
```

Šios dvi papildomos sąlygos yra vadinamos referencinio vientisumo suvaržymais, kadangi jos kontroliuoja, kaip valdomi susiję objektai. Sąlyga ON DELETE nurodo *SQL Server*, kad jei klientas pašalinamas iš duomenų bazės, šalinimo veiksmas turi nusiristi per visus susijusius užsakymus, tad pašalinami ir visi pašalinto kliento užsakymai. Sąlyga ON UPDATE nurodo *SQL Server*, kad jei *Klientų* (*Customers*) lentelėje pakeisime kliento *CustomerID*, naujasis ID turi būti įtraukiamas į visus to kliento užsakymus. Šios dvi sąlygos kartu užtikrina, kad pašalinus arba pakeitus klientą, nebūtų galima kurti „našlaičių“ užsakymų įrašų.

Egzistuoja kelios papildomos duomenų apibrėžimo komandos. Tarp jų yra komanda DROP (numesti), kuri pašalina lentelę iš duomenų bazės, ir komanda ALTER (keisti), kuri, apibrėžus lentelę, gali pakeisti jos struktūrą.

Dažniausiai naudojamas SQL komandų rinkinys klasifikuojamas kaip manipuliavimo duomenimis komandos arba manipuliavimo duomenimis kalba (DML). Šis rinkinys susideda iš tokių pagrindinių komandų:

- *Select* (pasirinkti)
- *Insert* (įterpti)
- *Delete* (pašalinti)
- *Update* (naujinti)

Komandą SELECT (pasirinkti) ištirsime išsamiai, nes tai sudėtingiausia iš manipuliavimo duomenimis komandų ir jos sintaksės elementai yra naudojami kitose komandose.

4.1.3 Manipuliavimas duomenimis: SELECT (pasirinkti)

Komanda SELECT (pasirinkti) naudojama, kai reikia išrinkti duomenis iš vienos ar kelių lentelių. Egzistuoja kelios šios komandos formos: nuo ganėtinai paprastos iki itin sudėtingos.

Pagrindinis pasirinkimas

Pagrindinė pasirinkimo komandos forma parenka visą lentelę, įskaitant visas eilutes ir stulpelius. Jos sintaksė yra tokia:

SELECT <išraiška> FROM <lentelė>

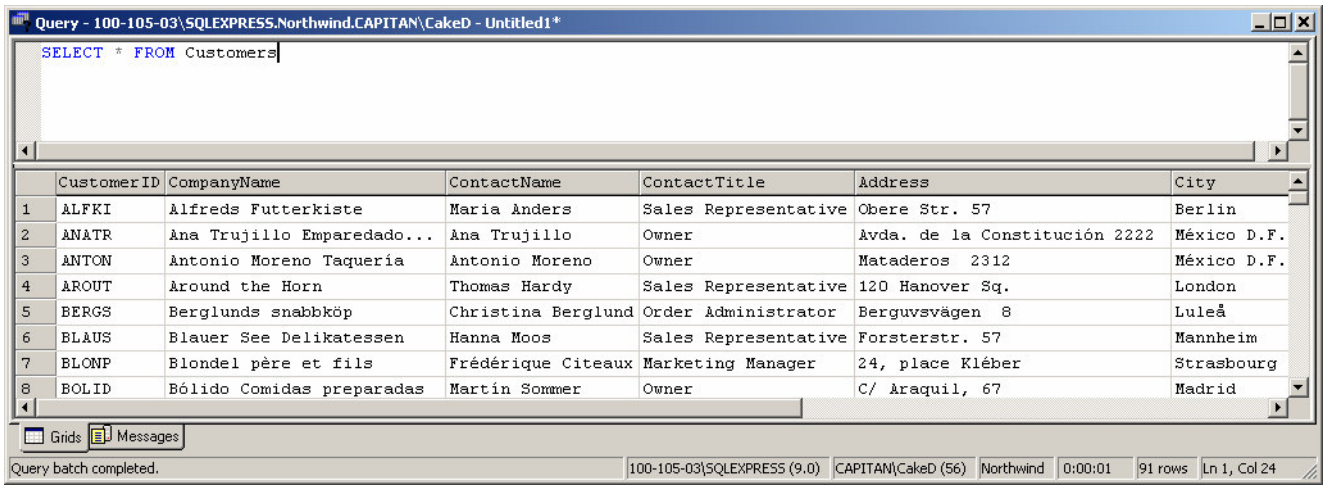
<lentelė> Tai duomenų bazėje esančios lentelės pavadinimas. Mūsų pavyzdinėje duomenų bazėje tai gali būti *Klientų* (*Clients*), *Produktų* (*Products*) ar *Užsakymų* (*Orders*) lentelės.

<išraiška> Pagrindinėje versijoje komandos SELECT (pasirinkti) išraiška atitinka „*“. Tai žvaigždutės simbolis, šiame kontekste reiškiantis „viską“.

Pavyzdžiui, norint pasirinkti visus įrašus iš *Klientų* (*Customers*) lentelės, reikėtų įvesti tokią komandą:

SELECT * FROM Customers

Šios užklausos rezultatas, kaip matoma programoje *SQL Server Query Analyser*, pateikiamas 4-3 pav. Šios užklausos rezultatas – tai visa *Klientų* (*Customer*) lentelė, įskaitant visas eilutes ir stulpelius.



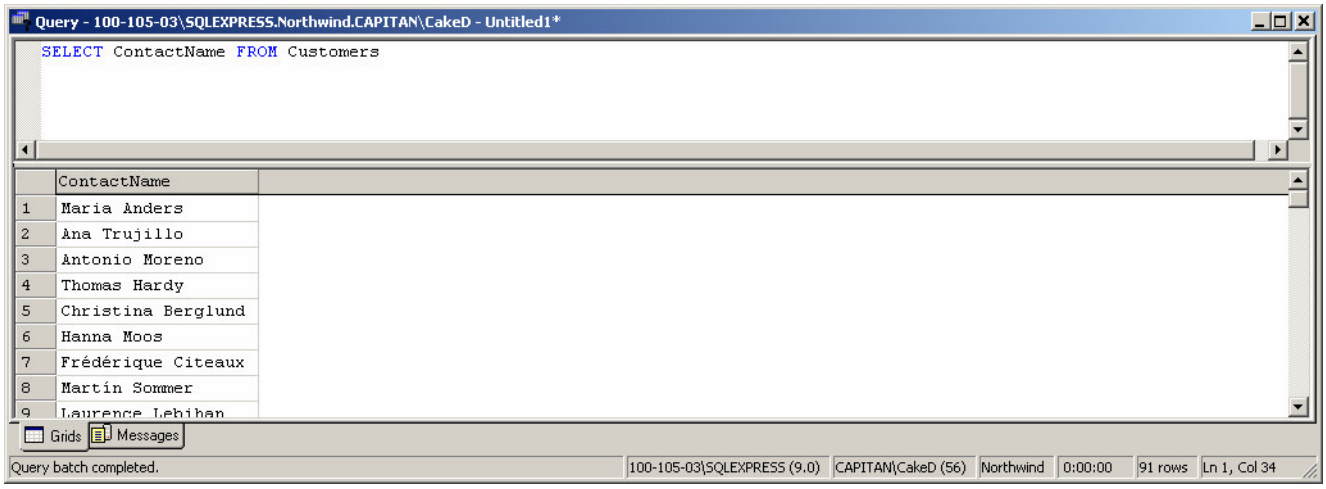
	CustomerID	CompanyName	ContactName	ContactTitle	Address	City
1	ALFKI	Alfreds Futterkiste	Maria Anders	Sales Representative	Obere Str. 57	Berlin
2	ANATR	Ana Trujillo Emparedado...	Ana Trujillo	Owner	Avda. de la Constitución 2222	México D.F.
3	ANTON	Antonio Moreno Taquería	Antonio Moreno	Owner	Mataderos 2312	México D.F.
4	AROUT	Around the Horn	Thomas Hardy	Sales Representative	120 Hanover Sq.	London
5	BERGS	Berglunds snabbköp	Christina Berglund	Order Administrator	Berguvsvägen 8	Luleå
6	BLAUS	Blauer See Delikatessen	Hanna Moos	Sales Representative	Forsterstr. 57	Mannheim
7	BLONP	Blondel père et fils	Frédérique Citeaux	Marketing Manager	24, place Kléber	Strasbourg
8	BOLID	Bólido Comidas preparadas	Martín Sommer	Owner	C/ Araquil, 67	Madrid

4-3 pav. Komandos SELECT užklausos rezultatai *SQL Server* aplinkoje

Galima adaptuoti šią bazinę formą ir ištirti daug įvairių dalykų. Galime pakeisti išraišką „*“ kita verte ir išanalizuoti tik *Klientų* lentelės stulpelį *ContactNames*:

```
SELECT ContactName FROM Customers
```

Ši komanda vis dar pateikia visas lentelės eilutes, tačiau rezultatuose matomas tik vienas stulpelis: *ContactName*. 4-4 pav. matote šios užklausos rezultatus.



	ContactName
1	Maria Anders
2	Ana Trujillo
3	Antonio Moreno
4	Thomas Hardy
5	Christina Berglund
6	Hanna Moos
7	Frédérique Citeaux
8	Martín Sommer
9	Laurence Lebihan

4-4 pav. Vieno stulpelio ištraukimas komanda SELECT

SELECT išraiškos gali pateikti įvairius dalykus, įskaitant visus stulpelius, parinktus stulpelius, lentelės eilučių skaičių, sumines vertes bei atlikti kitus aritmetinius veiksmus. Šios išraiškos ir SELECT komandų pavyzdžiai su šiomis išraiškomis pateikiami 4-1 lentelėje. Visos šios išraiškos grąžina arba tiria visas lentelės eilutes.

4-1 lentelė. SELECT komandos išraiškos

<išraiška>	Reikšmė	Pavyzdžiai
*	Pateikiama visa lentelė	Select * from Customers
count(*)	Pateikiamas lentelės įrašų skaičius	Select count(*) from Customers
Laukų pavadinimai	Pateikiamos užklaustų laukų vertės	Select Address from Suppliers Select ContactName, Phone from Customers
Skaičiavimai	Skaičiuojami skaitiniai laukai	Select UnitPrice + 1.14 from Products
Suminės vertės	Skaičiuojama duomenų bazės duotojo lauko ar visų laukų suma, vidurkis, minimumas arba maksimumas	Select avg(UnitPrice) from Products Select sum(UnitPrice) from Products Select max(UnitPrice) from Products Select min(UnitPrice) from Products

Eilučių apribojimas

Kol kas mums pavyko ištirti tam tikrus lentelės stulpelius, tačiau bazinė komanda SELECT negali apriboti rezultatų iki dominančių eilučių. Norėdami tai atlikti, įtrauksime į SELECT komandą naują sąlygą: WHERE (kur). Komandos SELECT su sąlyga WHERE sintaksė atrodo taip:

SELECT <išraiška> FROM <lentelė>
WHERE <sąlyga>

<lentelė> Tai duomenų bazėje esančios lentelės pavadinimas. Mūsų pavyzdinėje duomenų bazėje tai gali būti *Klientų* (*Clients*), *Produktų* (*Products*) ar *Užsakymų* (*Orders*) lentelės.

<išraiška> Gali būti vienas iš daugelio dalykų, apibrėžtų pirmiau pateikiamoje 4-1 lentelėje.

<sąlyga> Kai kurios sąlygos, ribojančios tiriamas eilutes. Kai konkretaus įrašo sąlyga patenkinama, tas įrašas pasirenkamas.

Pavyzdžiui, jei mes norėtume išrinkti tik tuos užsakymus, kurie buvo pateikti 1997 m. sausio 1 d., galėtume suformuoti tokią užklausą:

SELECT * FROM Orders WHERE OrderDate = 'Jan 1, 1997'

Šiuo atveju išraiška „*“ nurodo, kad mes norime pamatyti visus *Užsakymų* lentelės stulpelius, tačiau sąlyga WHERE apriboja rodomas eilutes. Sąlyga WHERE gali būti labai sudėtinga, analizuoti daug laukų ir įtraukti kelis kriterijus bei lyginamąsias vertes.

Atminkite, kad segmentas *'Jan 1, 1997'* išskiriamas apostrofais. Jie vadinama **skyrikliais**, o vertės priklausomai nuo išraiškos tipo yra atskiriamos skirtingai. Skaičiai skyrikliais neatskiriami, pvz.: Amount = 10. Simbolių segmentai ir datos (žr. pirmiau) atskiriami apostrofais. Simbolių segmento išraiška gali atrodyti taip: ContactName = 'Tom Jones'.

Kadangi skyrikliai dar kartą nurodo duomenų tipus, kurie skirtingose SDBVS programose gali būti nevienodi, tai – kitas SQL dialektų nevienodumo pavyzdys. Pavyzdžiui, programoje *MS Access* teksto segmentai turi būti imami į kabutes (“Tom Jones”), o datos ribojamos grotelių simboliais (#Jan 1, 1997#).

Ši konkreti užklausa lauke *OrderDate* surastoms vertėms su datos išraiška ‘Jan 1, 1997’ lyginti naudoja lygybės (=) **lyginimo operatorių**. Lyginimo operatoriai paprasčiausiai leidžia palyginti dvi vertes. Palaikomi lyginimo operatoriai pateikti 4-2 lentelėje.

4-2 lentelė. Lyginimo operatoriai

Simbolis	Reikšmė
=	Lygu
<	Mažiau
<=	Mažiau arba lygu
>	Daugiau
>=	Daugiau arba lygu
<>	Nelygu (kai kuriose įdiegtyse naudojama „!=“)

Šie operatoriai turi intuityvią reikšmę lyginant skaitines vertes, o palyginimai „lygu“ ir „nelygu“ naudojami (kaip ir tikimasi) dirbant su teksto segmentais ir datomis. Dirbant su datos vertėmis, „less than“ (mažiau) yra interpretuojamas kaip „prior to“ (iki), o „greater than“ (daugiau) – kaip „after“ (po). Pavyzdžiui, tolesnė užklausa pateiks visus *Užsakymus*, padarytus iki 1997-01-01.

```
SELECT * FROM Orders WHERE OrderDate < ’Jan 1, 1997’
```

Operatoriai „mažiau“ ir „daugiau“ su teksto segmentais nesielgia intuityviai. Teksto segmentų palyginimas naudojant operatorius <, >, >= ir <= atliekamas pagal Amerikos standartinio informacijos mainų kodo (*American Standard Code for Information Interchange*, ASCII) vertes. Kiekvienas simbolis turi priskirtą skaitinį kodą ir būtent šios skaitinės vertės ir yra lyginamos. Bendruoju atveju ASCII kodai yra nuoseklūs, tad „A“ suteiktas 65, o „B“ – 66, tačiau didžiosios ir mažosios raidės čia koduojamos skirtingai („a“ suteiktas 97). Rezultatas: ASCII sistemoje „BBB“ eina pirmiau už „aaa“.

Skaitinėms ir datos vertėms siūlomas papildomas lyginimo operatorius: BETWEEN (tarp). Jis leidžia teikti užklausas ieškant verčių tarp dviejų skaitinių konstantų. Tai logiškai yra tas pats, kas pirmiau pateikti aritmetiniai palyginimai, tačiau užklausa yra trumpesnė ir geriau skaitoma. Tarkime, kad mes pageidaujame rasti visus produktus, kurių kaina – nuo 5 iki 10 dolerių. Šiuo atveju galima naudoti abi užklausas:

```
SELECT * FROM Products WHERE UnitPrice BETWEEN 5 and 10
```

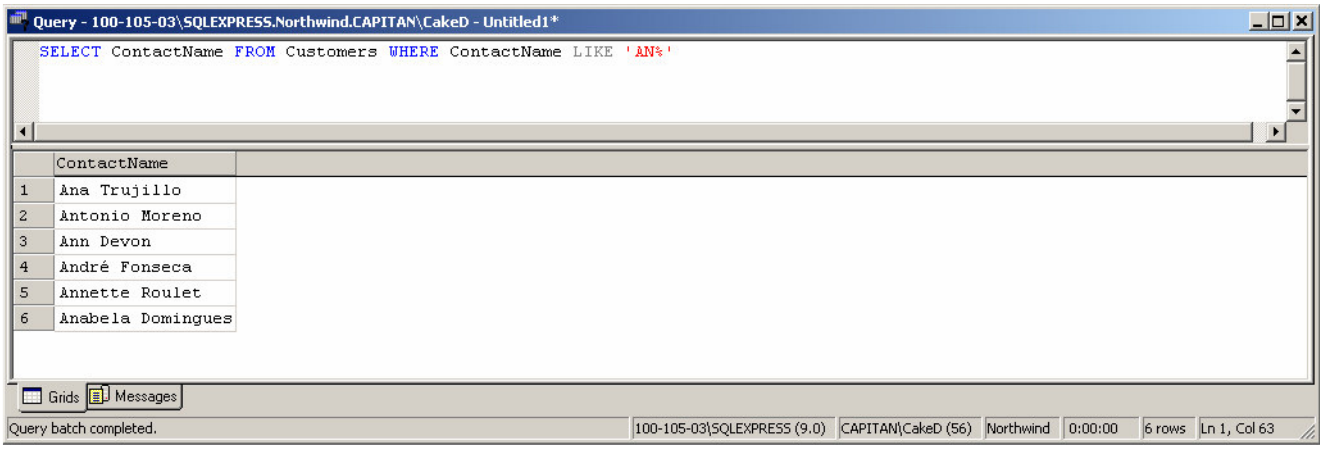
```
SELECT * FROM Products WHERE UnitPrice >=5 AND UnitPrice <=10
```

Atkreipkite dėmesį, kad pastarajame pavyzdyje du lyginimai sujungti **loginiu operatoriumi** AND (ir). Tai reiškia, kad, norint, kad įrašas būtų ištrauktas, abu šiek lyginimai turi būti TRUE (tiesa). Iš viso yra trys loginiai operatoriai: AND (ir), OR (arba) ir NOT (ne).

Teksto vertėms yra papildomas lyginimo operatorius, kuris leidžia palyginti teksto segmentus naudojant specialiuosius simbolius. Operatorius LIKE teksto segmentų struktūrai nustatyti naudoja specialųjį simbolį „%“. Šis specialusis simbolis reiškia „nė vieno arba daugiau simbolių“. ‘%Thomas’ reikštų bet kokį simbolių, einančių prieš „Thomas“, skaičių. Taigi, rezultatas būtų: „Thomas“ arba „Robert Thomas“, tačiau ne „Thomas Hardy“. Atminkite, kad LIKE operatoriui nėra skirtumo tarp didžiųjų ir mažųjų raidžių, tad lyginti su ‘%Thomas%’ yra tas pats, kas lyginti su ‘%thomas%’. Toliau pateikiamas LIKE operatoriaus naudojimo pavyzdys:

```
SELECT * FROM Customers WHERE ContactName LIKE ‘AN%’
```

Šios užklausos rezultatai pateikiami 4-5 pav.:



4-5 pav. LIKE operatoriaus naudojimo rezultatai

Prijungimai

Prijungimas – tai jungtis tarp dviejų skirtingų lentelių įrašų. „ArcMap“ naudojantys studentai galbūt jau naudojo prijungimus, kai reikėjo sujungti įrašus neerdvinėje lentelėje su įrašais geoobjektų klasėje, remiantis bendromis dviejų lentelių laukų vertėmis. Tai – pirminio ir išorinio raktų ryšys, kurį aptarėme 2 dalyje, įgyvendinimo temoje.

Prijungimą naudosime išrinkdami duomenis iš kelių lentelių vienu metu. Pavyzdžiui, klasės diagramoje 4-2 pav. egzistuoja ryšys tarp *Produktų* (*Products*) ir *Tiekėjų* (*Suppliers*). Ši 1:M (vienas su daugeliu) asociacija įgyvendinama įterpiant *Tiekėjų* lentelės pirminį raktą į *Produktų* lentelę, šioje jį paverčiant išoriniu raktu. Siekiant išrinkti informaciją iš šių abiejų lentelių vienu metu, būtinai reikia teisingai sujungti (*connect*) produktus su jų atitinkamais tiekėjais, tam naudojant išorinio rakto mechanizmą.

Mes „sujungiame“ šias dvi lenteles, įtraukdami sąlygą į WHERE, kuri teigtų, kad vertės dviejuose laukuose (pirminio ir išorinio raktų) yra lygios. Komandos SELECT prijungimo sintaksė:

```
SELECT <išraiška>
 FROM <1lentelė>, <2lentelė>, <3lentelė>
 where <sąlyga>
```

Atminkite, kad FROM teiginio dalimi yra daugiau nei viena lentelė. Be to, WHERE (kur) turi apimti sąlygą, kad prijungti laukai yra lygūs. Pavyzdžiui, jei pageidautume išrinkti visus produktus ir kiekvieno produkto tiekėją, naudotume šią komandą:

```
SELECT * FROM Products, Suppliers
      WHERE Suppliers.SupplierID = Products.SupplierID
```

Sąyga *WHERE Suppliers.SupplierID = Products.SupplierID* užtikrina, kad su tinkama *Produktų* lentelės eilute būtų rodoma tinkama *Tiekėjų* lentelės eilutė. Atminkite, kad toliau pateikiamiems duomenų bazės laukams įvedamas naujas žymėjimas: *Suppliers.SupplierID.* Tai duomenų bazių programai nurodo, kad mes lentelėje *Tiekėjai* (*Suppliers*) ieškome lauko *SupplierID*. Mums reikalingas šis taškinis žymėjimas, nes laukas *SupplierID* yra abiejose lentelėse. Išoriniai raktai nebūtinai vadinami taip pat, kaip ir pirminiai (tokiu atveju taškinis žymėjimas būtų nebūtinas, nes laukų pavadinimai būtų unikalūs), tačiau paprastai duomenų bazės yra kuriamos pagal šiuos principus, kai abu laukai yra pavadinti identiškai.

Mes į sąlygą WHERE taip pat galime įtraukti ir kitus kriterijus, taip siekdami apriboti rezultatų eilutes ir užtikrinti, kad prijungimas būtų formuojamas teisingai. Kaip ir anksčiau, WHERE sąlygoms sujungti naudosime loginius operatorius AND (ir) ir OR (arba). Toliau pateikiama ganėtinai sudėtinga užklausa, aptarsime kiekvieną jos eilutę atskirai. Eilučių numeriai nėra užklausos dalis, jie reikalingi toliau pateikiamam aptarimui.

```
1    SELECT FirstName, Lastname, UnitPrice * Quantity, OrderDate
2     FROM Employees, Orders, [Order Details]
3     WHERE Employees.EmployeeID = Orders.EmployeeID
4     AND Orders.OrderID = [Order Details].OrderID
5     AND UnitPrice * Quantity > 5000
```

1 eilutėje nurodoma, kad turi būti ištraukiamas tik nedidelis pasirinktų stulpelių skaičius, šiuo atveju – 4 stulpeliai. Ši eilutė taip pat pateikia aritmetinio skaičiavimo pavyzdį: „UnitPrice * Quantity“. Čia „*“ reiškia daugybą, tad rezultatas – vieno vieneto kaina, apdauginta iš užsakyto kiekio. Ši išraiška gali būti vertinama kaip visa užsakymo suma. Galimi tokie aritmetiniai operatoriai:

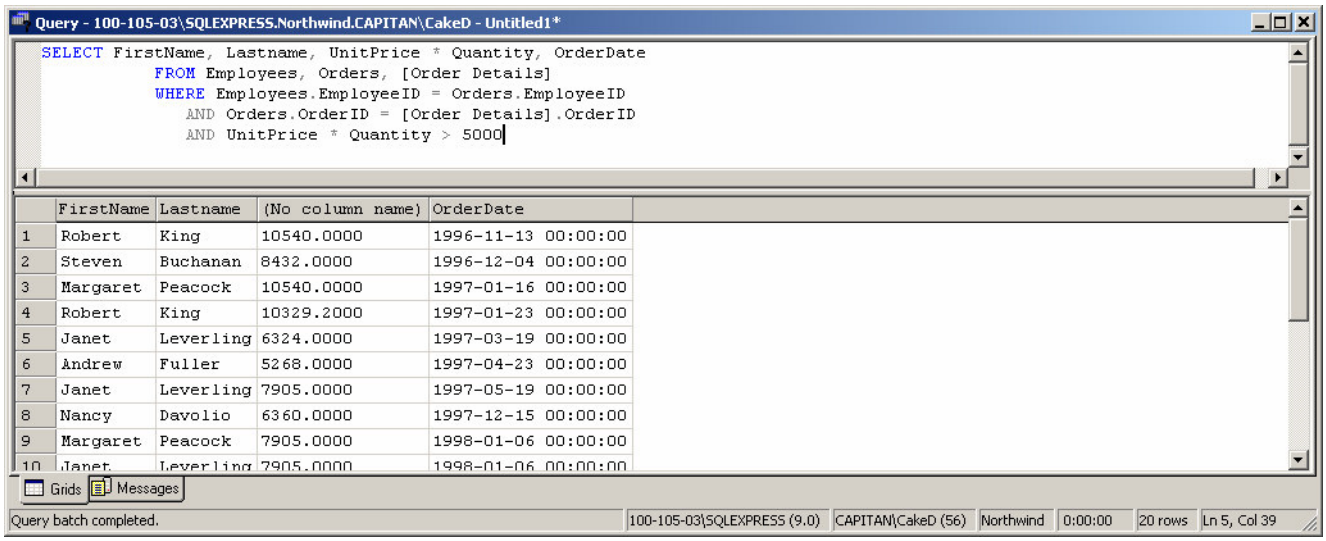
Simbolis	Reikšmė
+	Suma
-	Atimtis
*	Daugyba
/	Dalyba
%	Liekana (likęs sveikas skaičius, padalijus vieną skaičių iš kito, pvz., 5 % 2 = 1)
^	Laipsnis (pvz., 2 ^ 3 reiškia 2^3)

2 užklausos eilutė nurodo, kad reikalinga informacija pateikiama trijose atskirose lentelėse. Įtraukus tris lenteles, į užklausos dalį WHERE reikės įvesti bent dvi sąlygas: viena jų užtikrins, kad būtų tinkamai prijungti *Darbuotojai* (*Employee*) ir *Užsakymai* (*Order*), o antroji – kad būtų teisingai prijungti *Užsakymai* (*Orders*) ir *Užsakymų detalės* [*Order Details*]. Taip pat atkreipkite dėmesį, kad šioje eilutėje lentelė [*Order Details*] pateikiama laužtiniuose skliaustuose. Tai būtina, kadangi lentelės pavadinime yra tarpas. Laužtiniai skliausteliai reikalingi, kad SDBVS interpretuotų abu žodžius („*Order*“ ir „*Details*“) kaip vieną lentelės pavadinimą, o ne kaip du atskirus žodžius. Išsamumui pasiekti vartotojai gali į laužtinius skliaustus imti visus lentelių pavadinimus, tačiau juos būtina naudoti tik tuomet, kai lentelės pavadinime yra rezervuotas žodis (pvz., SELECT), kuris SDBVS reiškia ką kita arba kai lentelės pavadinime yra tarpų. *Northwind* duomenų bazėje šis lentelės pavadinimas su tarpu įtrauktas tam, kad studentams būtų pademonstruoti laužtinių skliaustų naudojimo principai.

Šios užklausos 3 ir 4 – tai apribojimai, užtikrinantys teisingą dviejų prijungimų atlikimą. Jie jungia išorinius ir pirminius raktus. Pastebėkite, kad čia vėl naudojamas taškinis žymėjimas, kadangi laukų pavadinimai nėra unikalūs.

5 eilutėje – papildoma sąlyga, apribojanti tiriamų eilučių skaičių. Čia mes norime matyti tik tas eilutes, kuriose bendra užsakymo vertė viršija 5 000 JAV dol.

Visos užklausos rezultatai pateikiami 4-6 pav.



4-6 pav. SELECT užklausos su dviem prijungimais rezultatai

Egzistuoja daug papildomų SELECT komandos tobulinimo variantų, pvz., rūšiavimas, kaupimas ir apibendrinančių lentelių (*pivot tables*) kūrimas. Jie yra vis sudėtingesni, tačiau, neturint praktinės patirties su pagrindinėmis SELECT komandomis, paliksime juos savarankiškam darbui.

4.1.4 Manipuliavimas duomenimis: INSERT (įterpti), UPDATE (naujinti) ir DELETE (šalinti)

Insert (įterpti)

Komanda INSERT leidžia vartotojams įtraukti duomenis į esamas duomenų bazės lenteles. Ši komanda įgauna tokią formą:

INSERT INTO <lentelė>
VALUES (<1atributas>, <2atributas>, …, <Natributas>)

Pavyzdžiui:

INSERT INTO Shippers
VALUES (4, 'Super Delivery', '(555) 241-2384')

Vertės turi būti imamos į skliaustus, jos turi būti tinkamai atskiriamos: kabutėmis atskiriami teksto segmentai ir datos, o skaičių papildomai atskirti nereikia. Kiekvienas įtraukiamas eilutės atributas atskiriamas kableliais.

Update (naujinti)

UPDATE <lentelė>
SET <1atributas> = <vertė>,
<2atributas> = <vertė>, t.t.

[WHERE <WHERE sąlyga>]

Pavyzdžiui:

```
UPDATE Customers
        SET Phone = '(250) 555-1234',
        ContactName = 'Wayne Gretzky'
        WHERE CustomerID = 12
```

Ši užklausa paprasčiausiai atnaujina vieno *Klientų* (*Customers*) lentelės kontakto vardą ir pavardę bei telefono numerį. Jei naujinami keli atributai, naujinimo sąlygos atskiriamos kableliais. Atminkite, kad komandų sintaksėje sąlyga WHERE yra neprivaloma. Jei ji praleidžiama, atnaujinamas kiekvienas lentelės įrašas.

Delete (pašalinti)

```
DELETE FROM <lentelė>
    WHERE <sąlyga>
```

Pavyzdžiui:

```
DELETE FROM Orders
        WHERE OrderDate < 'Jan 1, 1980'
```

Ši užklausa paprasčiausiai pašalina senesnius įrašus, kurie buvo įtraukti iki 1980 m. sausio 1 d.

4.1.5 Erdviniai SQL plėtiniai

Nors SQL komandos yra galingos, jos ribotos, nes palaiko tik paprastus duomenų tipus, pvz., skaičius, teksto eilutes ir datas. Erdvinėse duomenų bazėse turi būti numatyta galimybė išrinkti informaciją naudojant sudėtingesnius duomenų tipus, pvz., taškai, tiesės ir daugiakampiai (poligonai) bei sudėtingesnius ryšius, pvz., gretimumas ir artumas.

Pripažįstant erdvinių duomenų ir GIS pritaikymo skvarbą, tarptautiniai SQL standartai ir pagrindiniai SDBVS programų kūrėjai parengė standartinės SQL plėtinius, siekdami įtraukti erdvinių duomenų valdymo galimybę. Erdvinių duomenų naudojimą SDBVS standartizavo ir OGC, ir ISO. OGC nuoroda yra : *OpenGIS Implementation Specification for Geographic information - Simple feature access - Part 2: SQL option* (Geografinės informacijos OpenGIS įgyvendinimo specifikacija – Paprastų geoobjektų prieiga – 2 dalis: SQL parinktis). ISO nuoroda: *ISO/IEC 13249-3 SQL multimedia and application packages - Part 3: Spatial* (ISO/IEC 13249-3 SQL daugialypės terpės ir taikymo paketai - 3 dalis: erdviniai duomenys). ISO SQL daugialypės terpės standartas bendrai vadinamas SQL/MM. SDBVS erdvinės įdiegtys apima *Oracle Spatial*, *Informix Spatial Datablade* ir *DB2 Spatial Extender*. *SQL Server* atveju pilną erdvinį palaikymą planuojama įtraukti 2008 m. laidoje.

Abi – georeliacinė ir objektų reliacinė – struktūros turi apribojimų, susijusių su SDBVS naudojimu. Georeliacinė struktūra, jei pamenate, objektų atributus saugo reliacinėje lentelėje, o geometrijos duomenis – atskirame dvejetainiame faile. Daugelis objektų reliacinių įdiegčių visus GIS duomenis saugo vienoje reliacinėje duomenų bazėje, tačiau geometrijos duomenys laikomi BLOB (dvejetainiame stambiame objekte, *binary large object*) duomenų tipuose, koduotąja forma. Abi šios įdiegtys – kadangi jos erdvinių duomenų vaizdavimą grindžia skirtingais būdais – labai remiasi GIS programomis, kurios turėtų valdyti jų duomenis ir manipuliuoti jais, kadangi pati SDBVS nežino apie erdvinių objektų semantiką ar jų kodavimą.

Erdvinių duomenų saugojimo būdo standartizavimas – tai erdvinių standartų duomenų bazėms ir SQL įkūrimo pagrindas. Tai padarius pašalinama erdvinių ryšių tvarkymo programos lygiu būtinybė. Daugelis geografinių apdorojimo funkcijų bei didžioji dalis duomenų vientisumo kontrolės veiksmų gali būti perduodami pačiai SDBVS, ženkliai sumažinant GIS programai keliamus reikalavimus. Be to, standartizuotas saugojimo metodas įvairioms GIS programoms leidžia pasiekti tuos pačius duomenis be būtinybės konvertuoti.

Kalbant apie pagrindinę SQL, egzistuoja erdvinės SQL versijos dialektai, kadangi atskirų SDBVS kūrėjų įdiegtys šiek tiek skiriasi. Toliau pateikiamuose pavyzdžiuose naudojamas bendrinis dialektas, grindžiamas 1999 OGIS specifikacija ir taikomas keliuose literatūros šaltiniuose (pvz., Shekhar, 2003). Vartotojai gali nurodyti savo SDBVS dokumentaciją, siekdami apibrėžti tikslias dialekto subtilybes.

Erdvinė DDL erdvinių tipų naudojimo požiūriu yra labai panaši į standartinę SQL:

```
CREATE TABLE Cities (
        Name                            varchar(30),
        Population  int,
        Shape                   POINT
)

        INSERT INTO Cities VALUES (
                        'Some Little Town',
                        10784,
                        NEW POINT (493940.1, 5363403.2)
```

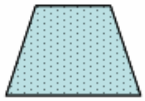
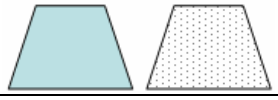
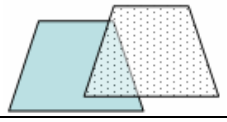
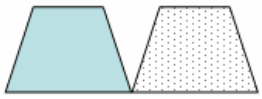
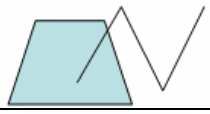
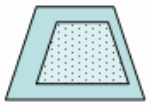
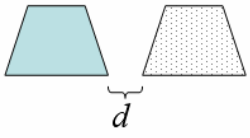
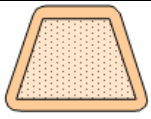
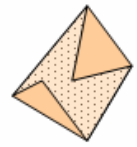
Atkreipkite dėmesį į erdvinių duomenų tipą POINT (taškas). Kiti duomenų tipai egzistuoja ir tiesiniams bei daugiakampiams objektams. Šie erdviniai duomenų tipai turi būti įtraukiami į SQL duomenų apibrėžimo kalbos dalis, o erdviniai skaičiavimai ir ryšiai – į manipuliavimo duomenimis komandas.

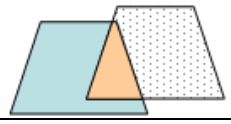
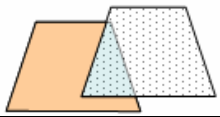
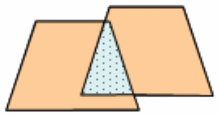
Pavyzdžiui, standartiniai SQL aritmetinių operacijų operatoriai, pvz., suma, atimtis, daugyba ir dalyba SQL/MM įdiegtyje turi apimti ir operatorius, skirtus atlikti erdvinius skaičiavimus. Siekiant leisti vykdyti tam tikrus skaičiavimus (pvz., riba ar linijos ilgis, vidaus plotas ir tiesinė kryptis), į SQL įtraukiami nauji operatoriai. Pavyzdžiui, toliau pateikiamoje užklausoje naudojamas operatorius AREA (plotas) kartu su erdviniu lauku *Shape* (forma) ir nustatomas kiekvienos valstybės teritorijos plotas.

```
SELECT Name, AREA (Shape)
    FROM Country
```

Jei standartinė SQL leidžia palyginti vertes naudojant lyginimo operatorius (daugiau, lygu, nelygu), SQL/MM turi apimti kelis naujus lyginimo operatorius, kurie atliktų erdvinius palyginimus. Naujieji operatoriai – tai: TOUCH (liesti), CROSS (kirsti), OVERLAP (perdengti), INSIDE (vidus), COVERS (dengia), COVERED BY (dengia (kas)), EQUAL (lygu) ir DISJOINT (išardyti). 4-3 lentelėje pateikiama šių operatorių santrauka.

4-3 pav. Topologiniai erdvinės SQL operatoriai

Operatoriaus tipas	Operatorius	Aprašas	Pavyzdys
Topologiniai operatoriai	*Equal* (lygu)	Jei dvi figūros yra vienodos, pateikia vertę TRUE (tiesa)	
	Disjoint (išardyti)	Pateikia vertę TRUE (tiesa), jei dvi figūros nesikerta (t.y. jos erdvėje atskirtos)	
	Overlap (perdengti)	Pateikia vertę TRUE (tiesa), jei susikerta vidinės dviejų geometrinių figūrų dalys	
	Touch (liesti)	Pateikia vertę TRUE (tiesa), jei susikerta dviejų figūrų ribos, tačiau nesikerta vidus	
	Cross (kirsti)	Pateikia vertę TRUE (tiesa), jei paviršius kerta kreivę	
	Contains (apima)	Pateikia vertę TRUE (tiesa), jei visa geometrinė figūra patenka į kitos geometrinės figūros vidų	
	Intersect (perkirsti)	Pateikia vertę TRUE (tiesa), jei dvi geometrijos nėra išardytos	Perdanga arba lietimas
Erdvinė analizė	*Distance* (atstumas)	Pateikia trumpiausią atstumą tarp dviejų figūrų	
	Buffer (buferis)	Pateikia figūrą, kuri susideda iš tam tikro ploto, duotuoju atstumu supančio duotąją figūrą	
	ConvexHull (išgaubta išaiža)	Pateikia figūrą apimančią mažiausią geometrinį išgaubtą rinkinį	

	Intersection (sankirta)	Pateikia dviejų geometrinių figūrų sankirtą	
	Union (sąjunga)	Pateikia dviejų geometrinių figūrų sąjungą	
	Difference (skirtumas)	Pateikia pirmosios figūros dalį, kuri nesikerta su antrąja figūra	
	SymDiff (sim. skirtumas)	Pateikia abiejų figūrų dalis, kurios nesikerta tarpusavyje	

Pavyzdžiui, apsvarstykime tris objektų klases: *Miestas* (*City*), *Upė* (*River*) ir *Valstybė* (*Country*). Galima parengti daug užklausų, kurios ištirtų erdvinius ryšius tarp objektų.

Iš visų Upių (River) lentelės upių reikia rasti valstybes, kuriomis jos teka:

```
SELECT Country.Name, River.Name
    FROM Country, River
    WHERE CROSS (River.Shape, Country.Shape)
```

Iš visų miestų reikia rasti tuos, kurie yra 300 m atstumu nuo Reino upės:

```
SELECT City.Name
    FROM City, River
    WHERE OVERLAP (City.Shape, BUFFER (River.Shape, 300))
            AND River.Name = 'Rhine'
```

Reikia rasti upių ilgius kiekvienoje valstybėje, kuria jos teka:

```
SELECT      River.Name,
            Country.Name,
            LENGTH ( INTERSECTION (River.Shape, Country.Shape))
    FROM River, Country
    WHERE CROSS (River.Shape, Country.Shape)
```

SQL plėtiniai erdviniams duomenims apdoroti yra ganėtinai nauji (adaptuoti 1999 m. SQL standartui). Pokyčiai vyksta itin sparčiai: apie juos liudija faktas, kad *SQL Server* jau pradėjo taikyti paramą erdviniams tipams ir erdvinėms funkcijoms. Tikimasi, kad nors šiuo metu egzistuoja tam tikrų erdvinės SQL apribojimų, pvz., tokių objektų kaip rastrai, trimačiai reljefo paviršiai, metaduomenys ir tinklai, tačiau parama erdviniams duomenims ir jų apdorojimui taps platesne vyraujančios SDBVS technologijos dalimi.

Su keliomis palaikomomis SDBVS *ArcGIS* vartotojai erdvinių objektų pateikimui gali rinktis iš BLOB arba standartizuotų geometrinių duomenų tipų. Standartizuotų duomenų tipų naudojimas reiškia, kad programos, nenaudojančios „ArcObjects“ sistemos, vis tiek gali dirbti su pagrindine duomenų baze, o trečiųjų šalių programos – pasiekti duomenis. Visose „ArcMap“ versijose iki 9.2 vartotojai buvo verčiami naudoti BLOB laukus koduoti dvejetaines geoobjektų geometrijos išraiškas. Tai – didelis žingsnis atviresnės GIS architektūros link ir svarbesnis pagrindinės DBVS vaidmuo GIS pritaikymo funkcijų atžvilgiu.

GIS programos vis dar remiasi principu didžiąją dalį darbų atlikti su geoduomenų baze. Jei geoduomenų bazės įrašais manipuliuoti SQL yra naudojama tiesiogiai, reikia imtis atsargumo priemonių, kadangi duomenys GIS programai gali tapti netinkami naudoti. SQL vartotojams, modifikuojantiems geoduomenų bazės informaciją, ESRI siūlo laikytis toliau išdėstytų gairių:

- Vengti keisti objektų klases, kurioms priskirtos tam tikros versijos (tik SDE geoduomenų bazėms).
- Po bet kurių modifikavimų SQL priemonėmis, naudoti patvirtinimo (*commit*) arba atšaukimo (*rollback*) operacijas, kadangi to nepadarius gali kilti keblumų. Pavyzdžiui, duomenų glaudinimo veiksmas operacijos patvirtinimo ar atšaukimo gali laukti neribotą laiką.
- Vengti keisti atributus, kurie per geoduomenų bazės elgseną veikia kitus duomenų bazės objektus (pvz., su geoobjektais susijusios anotacijos arba ryšių klasės).
- Vengti taikyti SQL keičiant geoobjektų klasės, kuri priklauso pažangiems geoduomenų bazės objektams arba funkcijoms, pvz., geometriniai tinklai, topologijos ir ryšiai, geometrines figūras.

4.2 Erdvinis indeksavimas

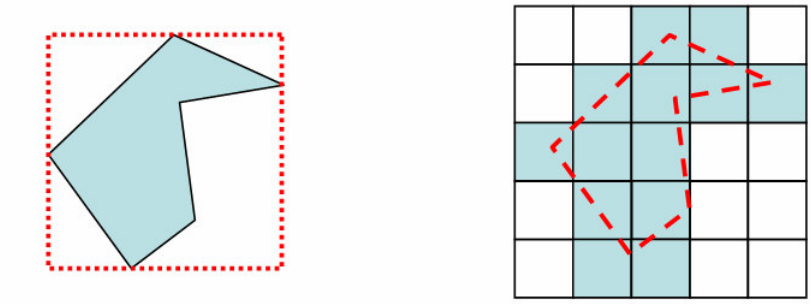
Indeksas – tai duomenų struktūra, leidžianti lengviau pasiekti duomenų objektus. Ji veikia tokiu pačiu būdu, kaip ir knygos rodyklė. Jei, pavyzdžiui, atitinkamame duomenų bazių vadovėlyje ieškote visų su SQL susijusių temų, neskaitysite dėl to visos knygos ir neieškosite segmento „SQL“. Jums užteks surasti SQL rodyklėje, atsiversti atitinkamus puslapius ir perskaityti tik juos. Indeksai neerdvinėje duomenų bazėje gali reaguoti itin sparčiai, net jei tenka analizuoti didelius duomenų kiekius.

Knygos rodyklė arba rodyklė, padedanti rasti banko sąskaitų numerius bankininkystės duomenų bazėje, tai – vienmačio indekso pavyzdžiai. Jie yra vienmačiai, nes atitinka vertes vienoje ašyje, pvz., visi žodžiai, surūšiuoti nuo A iki Z, arba banko sąskaitos, surūšiuotos skaičių didėjančia tvarka. Indeksas išilgai tos vienos ašies kontinuumo įgauna vertę ir jį duomenų bazėje tampa lengva rasti. Egzistuoja daug metodų, padedančių paspartinti vienmatę paiešką.

Erdviniai duomenys turi naudoti dvimačius arba trimačius indeksus, kadangi vieta koduojama bent dviem (x ir y) koordinatėmis, o neretai – ir trimis (x, y ir z). Dėl šios priežasties reikia taikyti kitokius metodus ir atstovauti dvimatę tiesinio pobūdžio erdvę bei pasitelkti esamus vienmačius metodus arba indeksavimui dvimatėje arba trimatėje erdvėje išvesti naujus metodus. Šioje temoje išsamiau nagrinėsime pastarąją situaciją, kadangi šie nauji metodai yra naudojami anksčiau mūsų aptartose SDBVS ir GIS programose.

Erdvinis indeksas veikia tokiu pat būdu, kaip ir vienmatis indeksas – jis saugo informaciją apie erdvinį objektų kontekstą, kad būtų galima greičiau rasti ir naudoti erdvinius objektus. Pamatinė erdvinio indeksavimo idėja – saugoti objekto geometrijos aproksimaciją. Ši supaprastinta geometrinė figūra tampa erdviniu raktu išgauti tikrąją objekto geometriją.

Vienas iš aproksimacijos tipų – tai **tolydžioji aproksimacija.** Tokios aproksimacijos yra grindžiamos pačių objektų koordinatėmis. Dažniausia jų – tai ribojantis rėmelis arba minimalus ribojantis stačiakampis (MBR), kuris yra mažiausias stačiakampis, lygiagretus koordinačių sistemos ašims ir visiškai apimantis tikrojo objekto geometrinę figūrą. Kitas aproksimacijos tipas – **matricos aproksimacija**. Šiuo atveju erdvė padalijama į taisyklingos matricos gardeles ir objekto geometrinė figūra būna atstovaujama rinkiniu gardelių, su kuriomis objektas susikerta. 4-7 pav. matote abiejų aproksimacijos rūšių pavyzdžius.



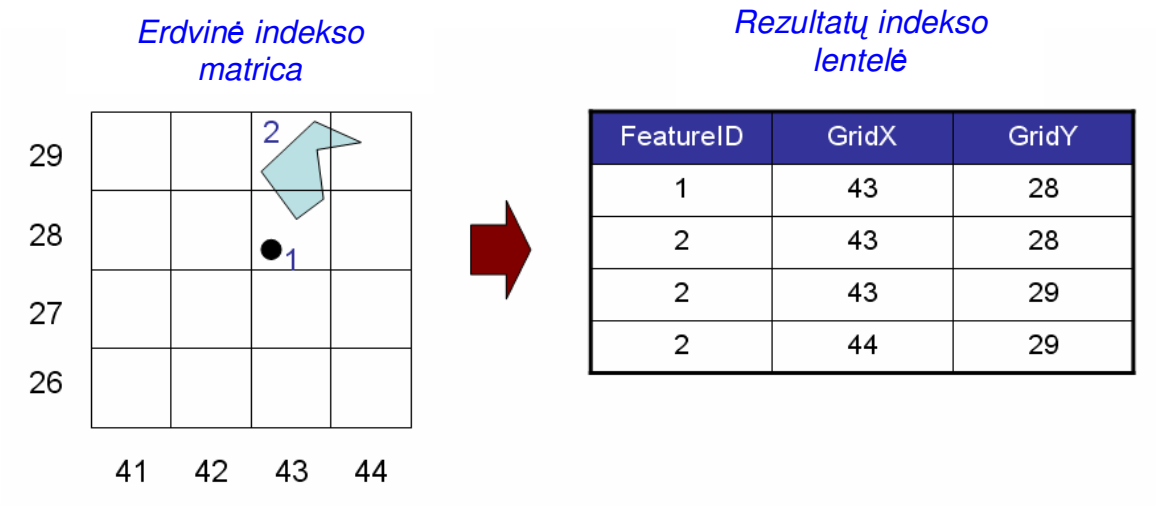
4-7 pav. Tos pačios objekto figūros MBR ir matricos aproksimacijos

Naudojant aproksimacijas kaip erdvinį raktą, užklausos apdorojimo metu objektams rasti pagal jų vietą taikomas dviejų etapų principas. Pirmojo etapo metu siekiama sumažinti išsamiai tirtinų objektų-kandidatų skaičių, šis etapas kartais vadinamas **filtravimu**. Tikimasi, kad filtravimo veiksmas greitai sumažins dominančių objektų skaičių, tiriant tik aproksimuotą geometriją. Likę galimi kandidatai, praėję filtravimo etapą, tiriami išsamiau. Tam taikomas užklausų apdorojimo **tobulinimo** veiksmas, kurį atliekant tikrinamos tikslios tikrosios geometrinės figūros koordinatės. Tokiu būdu, naudojant tik apibendrintas geometrines figūras, ištiriama didžioji dalis duomenų bazės objektų – daugeliu atvejų pats procesas gerokai paspartinamas. Išanalizuosime du erdvinio indeksavimo metodus: matricų indeksus ir R-medžio indeksus. Abu jie išnaudoja šiuos aproksimacijos metodus.

4.2.1 Matricos indeksas

Matricos indeksas gali būti supaprastintai įsivaizduojamas kaip kelių žemėlapio rodyklė. Vietoj 1-osios gatvės paieškos visame mieste skaitytojas gali ją surasti rodyklėje. Rodyklės įrašas gali jam nurodyti, kad 1-oji gatvė yra matricos koordinatėje A3. Tuomet skaitytojas peržvelgs eilučių ir stulpelių pavadinimus: eilutės galbūt bus pažymėtos raidėmis (A, B, C ir t.t.), o stulpeliai – skaičiais (1, 2, 3 ir t.t.). Toje vietoje, kur eilutė A susikirs su 3 stulpeliu, turėtų būti matricos gardelė, kurioje skaitytojas ras 1-ąją gatvę.

Funkcinių požiūriu taip GIS programoje veikia matricos indeksas, tai – matyt paprasčiausias būdas greitai nufiltruoti objektus-kandidatus. Siekiant pritaikyti šį metodą, programoje sukuriama matrica, kurioje gardelės pažymimos eilučių ir stulpelių indeksais (kaip pateikta 4-8 pav. kairėje). Tuomet sukuriama lentelė, kurioje išdėstoma sąveika tarp duomenų bazės geoobjektų ir šios matricos. Tokia lentelė pateikta 4-8 pav. dešinėje. Pavyzdžiui, erdvinėje pateiktyje „1“ pažymėtas taškas patenka į matricos gardelę, kurios indeksas yra: 43, 28. Indekso lentelėje esantis įrašas 1, 43, 28 reiškia, kad 1 geoobjektas yra 43, 28 gardelėje. Daugiakampis, pažymėtas „2“, yra trijose gardelėse, tad indekso lentelėje sukurtos trys eilutės, nurodančios, kad 2 geoobjektas kertasi su matricos gardelėmis.



4-8 pav. Paprastas matricos indekso mechanizmas

Svarstymo sumetimais įsivaizduokime, kad vartotojas suranda taško vietą žemėlapio lange ir nori nustatyti, kokį geoobjektą spustelėjo. Veiksmai, kurių tokiu atveju imasi GIS programa, galėtų atrodyti taip:

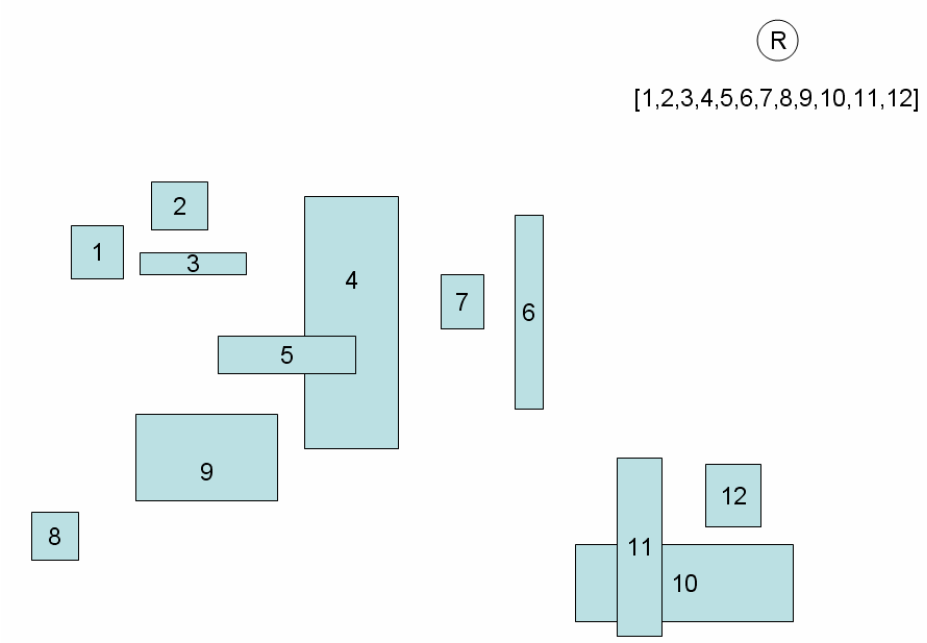
1. Nustatomos vartotojo spustelėtos matricos gardelės koordinatės *x* ir *y*. Tarkime, kad vartotojas spustelėjo gardelę 43, 28.
2. Peržvelgiama indekso lentelė ir randami visi matricos lentelės 43, 28 įrašai. Ši lentelė informuoja, kad geoobjektai 1 ir 2 kertasi su matricos gardele 43, 28. Tai – filtravimo veiksmas, kadangi greitai nustatėme, jog iš visų duomenų bazės geoobjektų yra du kandidatai, kuriuos *galėjo* spustelėti vartotojas.
3. Galiausiai programa, naudodama tobulinimo veiksmą, turi pažvelgti į tikrąją 1 bei 2 geoobjektų figūrų geometriją ir faktinę x bei y vietą, kurioje spustelėjo vartotojas ir nustatyti, kurį geoobjektą(-us) jis iš tikrųjų spustelėjo.

Šio proceso spartos esmė – ta, kad GIS programai iš pradžių nereikia peržvelgti visų duomenų bazėje esančių geoobjektų, nustatant, kuris geoobjektas buvo pasirinktas, o antra – didelei daliai geoobjektų nėra tiriama faktinė geometrinė geoobjektų figūra. Matricos veiksmingumas apsprendžiamas matricos gardelių dydžiu, kuris yra kompromisas tarp filtravimo veiksmingumo ir indekso lentelės apimties. Didelės matricos gardelės reiškia, kad duotoje gardelėje yra daug geoobjektų, tad po filtravimo etapo gali likti nemažai tirtinų geoobjektų, kurios reikės išanalizuoti tobulinimo etapo metu. Kai matricos gardelė didelė, indekso lentelė būna maža, tad ji nuskaitoma itin sparčiai. Atvirkščiai – kai matricos gardelė maža, filtravimo etapas yra itin veiksmingas ir gali dramatiškai sumažinti skaičių geoobjektų-kandidatų, perkeliamų į tobulinimo etapą. Tačiau tokiu atveju indekso lentelė gali tapti labai didele ir jos analizė gali užtrukti ganėtinai ilgai.

4.2.2 R-medžio indeksas

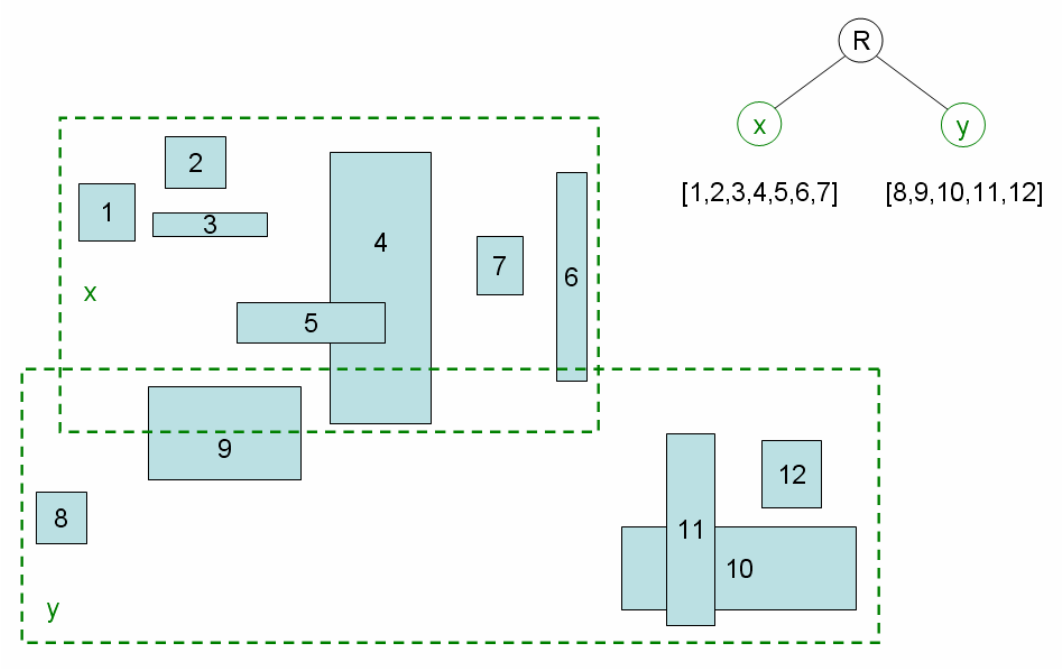
R-medžio indekso metodas išnaudoja minimaliai ribojamų daugiakampių įterptinius sluoksnius. Pagrindas – saugoti kiekvieno duomenų bazės geoobjekto MBR. Daugiakampiai ir tiesių objektai atstovaujami stačiakampiais, tuo tarpu taško MBR – tai paprasčiausias taškas. Pateiksime pavyzdžius su stačiakampiais MBR, tačiau tas pats R-

medžio procesas veikia ir su taškais. 4-9 pav. matote sunumeruotų stačiakampių seką, kiekvienas jų atstovauja duomenų bazės geoobjekto MBR. „R“ ir skaičiai viršutiniame dešiniajame kampe rodo šio išskaidymo lygio R-medžio pateiktį. Čia jį galima būtų lyginti su itin didele matricos gardele matricos indeksavimo metode, o visi MBR būtų laikomi R-medžio **šaknimis**, tad veiksmingai nevykdoma jokia erdvinė paieška. Šaknis – tai medžio struktūros pradžios taškas. Žymėjimai [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12] nurodo faktą, kad visi geoobjektų stačiakampiai yra nedalijamoje šakninėje koordinačių erdvėje arba ja yra apibrėžiami.



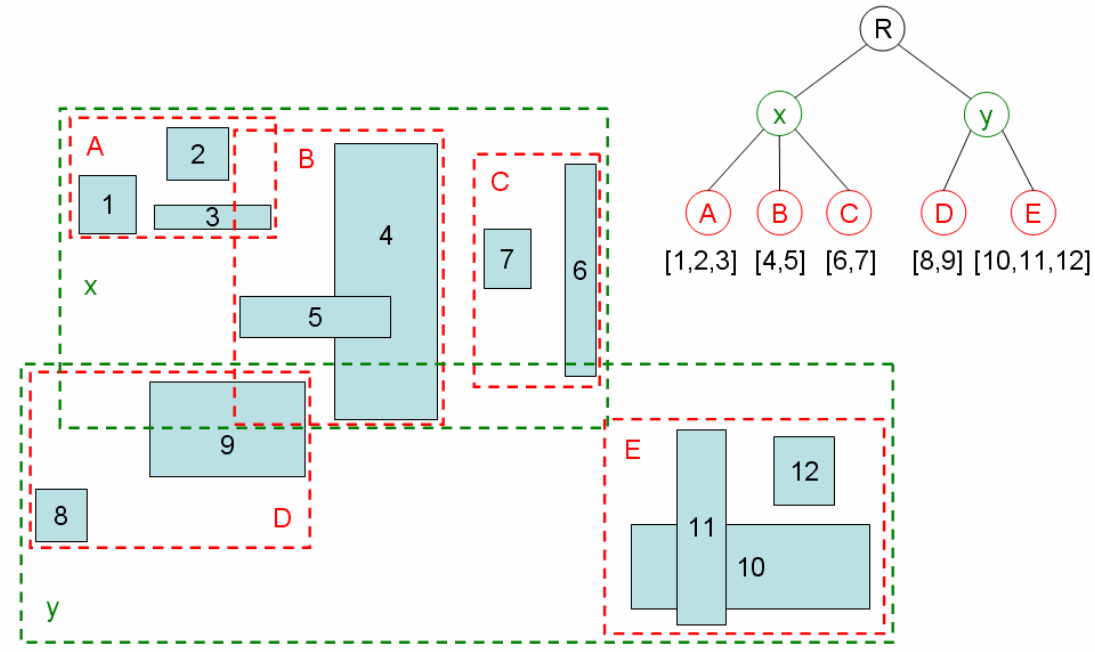
4-9 pav. Geoobjektų MBR atstovaujami taikant R-medžio indeksavimo metodą

Tada dvimatė erdvė yra skaidoma į kitą MBR rinkinį. Šiuo atveju naujieji MBR – tai minimalūs stačiakampiai, apimantys objekto MBR poaibį. 4-10 pav., pavyzdžiui, visi objekto MBR yra sugrupuoti į du naujus MBR, kurie apima objekto MBR. Taip dvimatė duomenų bazės erdvė suskaidoma į du MBR, žymimus *x* ir *y*.



4-10 pav. Dalinis R-medžio suskaidymas

Medžio iliustracija viršutinėje dešinėje dalyje dabar rodo, kad šaknis arba visa duomenų bazės koordinačių erdvė yra padalinta į du MBR. Pavyzdžiui, *MBR y* dabar apima 5 objektus (nuo 8 iki 12). *MBR x* ir *y* gali būti skaidomi toliau į mažesnius MBR, kaip parodyta 4-11 pav.



4-11 pav. Pilnas R-medžio suskaidymas

Čia, pavyzdžiui, *MBR Y* yra suskaidytas į du mažesnius stačiakampius, pažymėtus D ir E. Šiuo būdu kiekvieno stačiakampio užimama erdvė yra skaidoma į du mažesnius stačiakampius, apimančius objekto MBR, kurie į jį patenka. Medžio struktūra viršutiniame dešiniajame kampe dabar susideda iš trijų lygių, nurodant erdvės skaidymo seką.

Siekiant rasti duotąjį geoobjektą, yra ganėtinai nesunku peržvelgti medžio struktūrą ir jį aptikti. Pavyzdžiui, tarkime, kad vartotojas spustelėjo žemėlapį toje vietoje, kuri patenka į geoobjekto MBR pažymėtą „2“, netoli viršutinio kairiojo kampo. Paieškos procesas prasideda nuo R-medžio struktūros šaknų. Sistema palygina paiešką *x*, *y* su dviem antriniais šaknies stačiakampiais, pažymėtais *x* ir *y*. Spustelėtas taškas patenka į *x*, o ne į *y*, tad visą likusią medžio dalį po *y* galima atmesti. Tada spustelėtas taškas palyginamas su antriniais *x* stačiakampiais: A, B ir C. Randame, kad spustelėtas taškas patenka tik į MBR A, tad galima atmesti MBR B bei C. Stačiakampis A informuoja, kad jame yra trys geoobjektai: 1, 2 ir 3. Tuomet galima palyginti šių geoobjektų MBR su spustelėtu tašku. Tai atlikus, atrandame, kad vartotojas spustelėjo 2 geoobjektą. Norint sužinoti, ar spustelėtas taškas liečiasi su tikrąja 2 geoobjekto geometrine figūra, reikia atlikti detalų palyginimą (tobulinimo proceso dalis). MBR taip pat saugo 2 geoobjekto nuorodą arba raktą tikrajam duomenų bazės objektui rasti, kad būtų galima atlikti detalų geoobjekto geometrijos palyginimą.

Šio indeksavimo tipo spartos esmė – tai, kad iš analizės galima pašalinti dideles duomenų bazės dalis (filtravimas), nes paieškos metu toliau tiriamos tik aktualios medžio struktūros šakos. Be to, reikia išanalizuoti tik nedidelio skaičiaus geoobjektų tikrąją geometriją, mūsų atveju tai – tik vienas geoobjektas. Egzistuoja galimybė, kad persidengiančių MBR atveju (pvz., tam tikra ploto dalis yra bendra *x* ir *y*) reikės analizuoti kelias medžio šakas.

R-medžio struktūroje turi būti gerai subalansuotas aukštis, kad nuo šaknies būtų pašalinamas tas pats geoobjektų MBR gylis arba lygių skaičius. Visi geoobjektų MBR pateikiami tame pačiame medžio lygyje. Paprastai R-medžiai yra ganėtinai seklūs ir platūs. Pavyzdžiui, R-medis, indeksuojantis 100 mln. geoobjektų, gali būti 5 gylio lygių, o kiekvienoje „šakoje be lapų“ gali būti virš 100 antrinių geoobjektų (Shekhar ir Chawla, 2003. 101 p.). Šaka be lapų – tai vienas vidinis medžio lygis, pvz., *x* ir *y*.

R-medžio ieškos sparta priklauso nuo dviejų veiksnių:

Dengimas — Dengimas – tai duotojo lygio medžio ploto dalis, kurią apima antrinis MBR – netiesioginis kiekvieno lygio „tuščiojo ploto“ arba motininės šakos, kurios nedengia antrinė šaka, dalies matavimo būdas. Pavyzdžiui, antriniai stačiakampiai D ir E faktiškai dengia tik nedidelę dalį ploto, kurį apima motininis stačiakampis *y*. Šios šakos atveju dengimas yra ganėtinai mažas. Stačiakampio x dengimas palyginimui yra ganėtinai didelis, kadangi A, B ir C stačiakampiais uždengta žymi x dalis. Siekiant spartesnio R-medžio struktūros veikimo, palankiau yra iki minimumo sumažinti dengimą. Mažas dengimas reiškia, kad, analizuojant kiekvieną medžio šaką, galima atmesti didelius plotus.

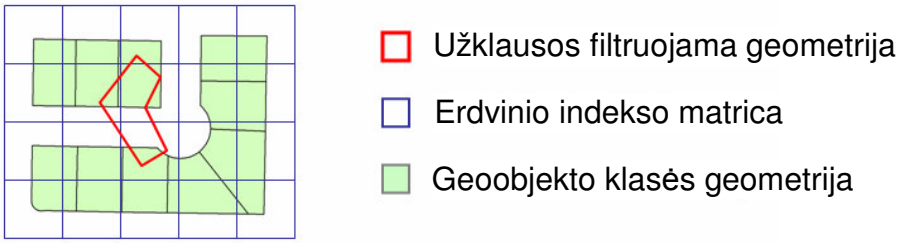
Perdanga — Perdanga – tai rodiklis, apibūdinantis, kiek stačiakampiai duotuoju lygiu persidengia vienas su kitu. Pavyzdžiui, stačiakampiai *x* ir *y* persidengia maždaug 10 proc. ploto, kurį jie abu padengia. A - E stačiakampiai greičiausiai persidengia dar mažesniu lygiu nei 10 proc. Iš spartos perspektyvos persidengimą taip pat norime sumažinti iki minimumo. Kai du stačiakampiai persidengia, egzistuoja tokia galimybė, kad į analizę reikės įtraukti abi antrines šakas.

Užduotis iki minimumo sumažinti perdangos rodiklį R-medžio spartos veiksmingumui yra aktualesnė nei sumažinti dengimą. Taigi, siekiant sumažinti perdangos rodiklį, mėginama daug kartų tobulinti R-medžio struktūrą. Pavyzdžiai; R*-medžio, R+-medžio ir supakuota R-medžio struktūros.

4.2.3 ESRI erdvinis indeksas

Erdviniai *ArcMap* indeksai naudoja įvairius mechanizmus, priklausomai nuo duomenų šaltinio. Asmeninės geoduomenų bazės išnaudoja viengubą matricos indeksą, tuo tarpu failų geoduomenų bazės ir *ArcSDE* geoduomenų bazės *Oracle*, *SQL Server* bei DB2 aplinkose naudoja sistemą, erdvinio indekso rėmuose apimančią iki trijų matricų. Erdviniai duomenys, saugomi *Oracle Spatial* ir *Informix ArcSDE* geoduomenų bazėse, naudoja R-medžio indeksą, kuris valdomas tiesiogiai SDBVS. Matricos indekso veiksmingumą apsprendžia naudojamos matricos gardelės dydis. Šį dydį gali apibrėžti vartotojas, tad, siekdami veiksmingiau valdyti erdvinius indeksus *ArcMap* aplinkoje, trumpai apibūdinsime, kaip veikia *ArcMap* matricos indeksai.

Svarstymo sumetimais darysime prielaidą, kad vartotojas mėgina nustatyti, kuriuos žemės sklypus kerta atsitiktinis daugiakampis geoobjektas. Mums reikia ieškomų daugiakampių (žemės sklypų) *geoobjektų klasės geometrijos* ir daugiakampio, kurį naudosime užklausai su *filtro geometrija* pateikti. Situacija gali atspindėti 4-12 pav. pateikiamą atvejį.

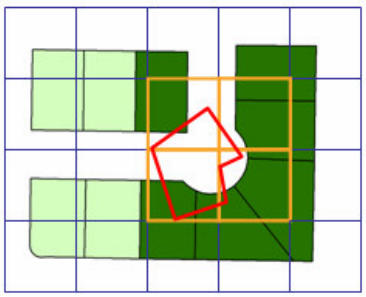


4-12 pav. Erdvinės užklausos pavyzdys

ESRI indekse veikia keturių pakopų filtras ir tobulinimo procesas (žr. toliau).

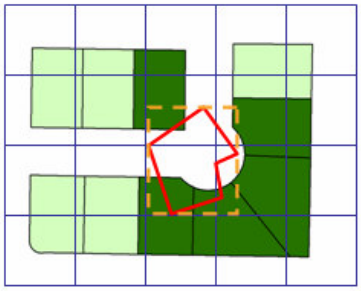
1. **Matricos išklotinė: filtravimas**. Šiuo veiksmu randami visi geoobjektų klasės įrašai, kuriuose filtruojamos geometrijos figūros kertamos matricos gardelės atitinka matricos gardeles, kurias kerta geoobjekto klasės geometrijos figūros. 4-13 pav. oranžine spalva vaizduojamos matricos gardelės, kurias kerta filtruojama geometrinė figūra. Šios gardelės savo ruožtu kerta kelis žemės sklypus. Sklypai-kandidatai vaizduojami tamsiai žalia spalva.

   Šis pirmasis filtravimo proceso etapas atmeta didžiausią dalį geoobjektų-kandidatų. Pateiktame pavyzdyje akivaizdžiai nėra iliustruojama ta didžioji neparinktų geoobjektų dalis, kadangi mes tikslingai padidinome nedidelę dominančią sritį. Likę geoobjektai-kandidatai pereina į 2 etapą.



4-13 pav. Likę kandidatai po 1 etapo

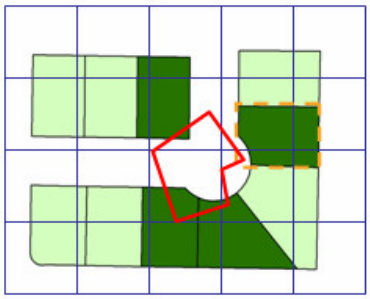
2. **MBR: filtravimas.** Antruoju etapu imami nufiltruotų geometrinių figūrų MBR arba *erdvinis apvalkalas* (*envelope* ESRI dokumentacijoje) ir lyginami su visų objektų-kandidatų MBR. Tie, kurių MBR kerta filtruojamos geometrinės figūros MBR, lieka kandidatais ir pereina į 3 etapą. 4-14 pav. pateikiamos filtruojamos geometrinės figūros MBR žemės sklypų atžvilgiu bei gaunami likę geoobjektai-kandidatai. Šio etapo metu kandidatų skaičius sumažinamas nedaug.



4-14 pav. Likę kandidatai po 2 etapo

Abu pirmesni etapai atliekami SDBVS lygiu, tad jie vyksta itin greitai. Likę kandidatai perduodami SDE (*spatial database engine*), kuris užbaigia užklausos apdorojimą.

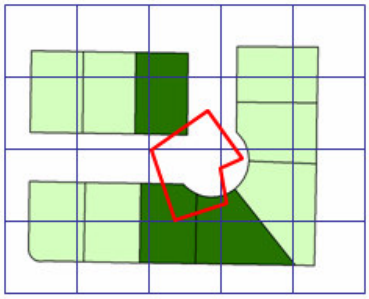
3. **Filtruojamos geometrinės figūros geoobjektų klasės *erdvinis apvalkalo* filtravimo metu**. Galutinis filtras, pritaikomas prieš atliekant pilną palyginimą su geoobjektų klasės geometrinėmis figūromis, palygina erdvinio filtro geometriją su likusių kandidatų MBR. 4-15 pav. matome likusius kandidatus 3 etapo pabaigoje. Oranžinė MBR viename įdubusiame sklype rodo, kodėl jis šiame etape išlieka kandidatu.



4-15 pav. Likę kandidatai po 3 etapo

4. **Filtruojamos geometrinės figūros geoobjektų klasės geometrijos tobulinimo metu**. Šiame paskutiniame etape skaičiavimai vykdomi intensyviausiai. Čia filtruojama geometrija lyginama su visa kiekvieno likusio kandidato geometrija. 4-16

pav. pateikiamas galutinis kirtimo operacijos rezultatas, užbaigus visus keturis apdorojimo etapus.



4-16 pav. Galutinis kirtimo rezultatas

4.3 Laiko duomenų saugojimas

GIS tapo veiksmingu erdvės valdymo ir analizės įrankiu. Tačiau gebėjimo tirti erdvinių objektų pokyčius laiko tėkmėje lyginant su neerdviniais objektais pažanga vis dar yra menka. Beveik visi žemėlapio elementai laikui bėgant gali kisti, ypač tokie vienetai kaip politinės ribos, augmenija ir žemės nuosavybė. Laiko elementas dar svarbesnis toms disciplinoms, kurioms pokyčiai laike nėra tik duomenų valdymo iššūkis (pvz., žemės nuosavybė), bet esminis tyrimo objektas. Tokios sritys – tai kriminalistika, ligų epidemiologija ir hidrologinė dinamika.

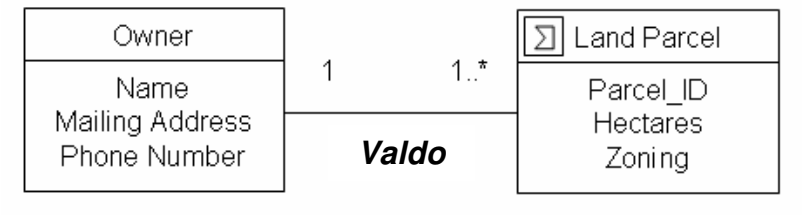
GIS technologijos šaknys yra kartografijoje ir statiškų žemėlapių kūrimo automatizacijos srityje. Tačiau GIS plėtra į daugelį skirtingų pritaikymo sričių, erdvinių duomenų pasiekiamumo augimas ir padidėję kompiuterių saugojimo bei analizavimo gebėjimai prisidėjo prie laiko kaip būtinojo geografinės analizės matmens svarbos didėjimo. Geografai seniai svarsto, kad geografinė informacija yra grindžiama trimis pagrindiniais komponentais: vieta, atributu ir laiku, tačiau tik XX a. IX dešimtmetyje laiko elemento tyrimai GIS aplinkoje tapo rimtesniais.

Pokyčio pobūdžiui apibrėžti galima naudoti įvairius terminus, o su GIS mes šias charakteristikas būtent ir stengiamės išgauti bei aprašyti. Pokyčių pobūdis gali būti aprašomas taip:

Tolydus	Pokytis tam tikrame laiko intervale vyksta ganėtinai pastoviu rodikliu (pvz., geologiniai arba hidrologiniai procesai).
Daugumos	Pokytis vyksta didžiąją laiko dalį, tuo tarpu rečiau pokytis nevyksta.
Atsitiktinis	Pokytis tam tikrame laikotarpyje vyksta su pertrūkiais, didžiąją laiko dalį išlikdamas statišku.
Unikalus	Pokytis įvyksta tik vieną kartą.

4.3.1 Laikas duomenų bazėje

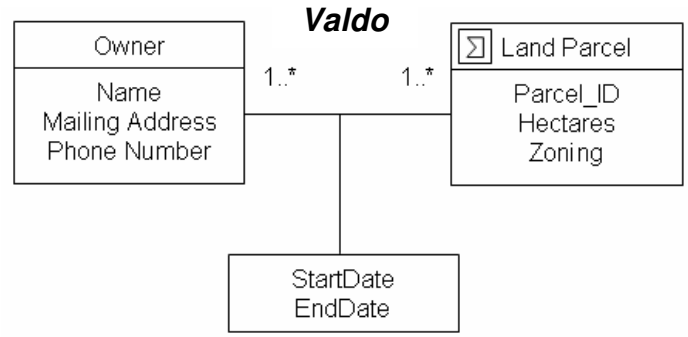
Neerdvinėse duomenų bazėse įrašų laiko priklausomybės klausimus galima spręsti, įtraukiant atributus, kurie apibrėžtų duotojo įrašo galiojimo laikotarpį. Daugelis laikinų reiškinių gali būti užfiksuoti pagal šią paprastą struktūrą, įtraukiant du atributus: *FromDate* (nuo) ir *ToDate* (iki). Šie du atributai nurodo laikotarpį, kuriuo galioja geoobjektas ar duomenų bazės eilutė. Toks požiūris ypač veiksmingas reiškiniams, kurie egzistuoja trumpai ir turi itin specifines susiformavimo bei išnykimo datas. Pavyzdys – *Darbuotojų* (*Employee*) lentelė, kurioje mes saugome asmens įdarbinimo bei atleidimo iš darbo datas. Pateikiant užklausą šiems dviem laukeliams, gauname informaciją, kas mums dirbo bet kuriuo momentu.
Galima išplėsti šią sąvoką toliau ir įvesti laikinus laukelius, turinčius asociacijas su mūsų duomenų baze. Apsvarstykite Savininkų-Žemės sklypų klasės diagramą, kurią aptarėme keliose šio kurso vietose. Ši diagrama dar kartą pateikiama 4-17 pav.



4-17 pav. Savininkų-Žemės sklypų diagrama – be laiko elemento

Šioje diagramoje modeliuojamas statiškas asociacijos požiūris. Čia duotasis žemės sklypas gali priklausyti tik vienam asmeniui, tad mes negalime modeliuoti nuosavybės istorijos laiko tėkmėje. Tai – laikinas statiškas modelis, kuris gali atspindėti tik vieną laiko momentą.

Kita vertus, jei pakeisime modelį ir leisime įtraukti daug savininkų, 1:M (vienas su daug) asociaciją pakeisime M:M (daug su daug) asociacija. Be to, paprastą M:M asociaciją paversime *asociacijos klase*, tad galėsime įrašyti pradžios ir pabaigos laikus kartu su asociacija. Tokiu būdu galėsime įrašyti kelis duotojo žemės sklypo savininkus ir laikotarpį, kuriuo jie nuosavybės teise turėjo konkretų žemės sklypą. 4-18 pav. vaizduojamas toks modelis.



4-18 pav. Svininkas-Sklypas su nuosavybės istorija

Kitas neerdvinės duomenų bazės pokyčių stebėjimo metodas – operacijų žurnalo naudojimas. Jis gali įgauti formą atskiros lentelės, kurioje būtų reziumuojami visi atlikti pakeitimai ir fiksuojamas laikas, kada jie buvo atlikti. Operacijų lentelėje gali būti tokia informacija:

```
CREATE TABLE TransactLog (
        TransID          Number (10)        NOT NULL,
        TableName        Varchar (30)       NOT NULL,
        RowID            VarChar (200)      NOT NULL,
        Description      Varchar (2000)     NOT NULL,
        TransDate        Date               NOT NULL
```

Šios lentelės struktūra leidžia išrinkti kiekvieną duomenų bazėje atliktą pokytį, tad gali būti itin naudinga, aprašant atliktų pokyčių tipus ir tikslų įvykių laiką. Tačiau ji neparodo išsamaus vaizdo, kaip duomenų bazė atrodė bet kuriuo duotuoju laiku. Pavyzdžiui, jei duomenų bazės įrašas buvo pašalintas, mes paprasčiausiai žinome, kada jis buvo pašalintas ir kuris įrašas buvo pašalintas. Mes nežinome, kokia informacija buvo laikoma

pašalintame įraše, tad šis metodas yra ribotas. Operacijų žurnalo požiūris yra labai veiksmingas, tačiau – tik nedidelio transakcijų tipo duomenų bazėse. Pavyzdžiui, banko sąskaitos balansą gali paveikti trijų tipų operacijos: pervedimas, indėlis ir išėmimas. Vedant paprastą operacijų žurnalą šių paprastų operacijų istorijai saugoti, galima atkurti sąskaitos balansą bet kuriuo metu.

Geografiniai atvejai yra daug sudėtingesni; objektai gali egzistuoti ribotą laiką (dėl ko reikia įtraukti atributus *FromDate* (nuo) ir *ToDate* (iki)), tačiau jie taip pat gali judėti, keisti formą ir atributus. Pavyzdžiui, laikui bėgant gali būti kaupiami arba dalijami žemės sklypai, tačiau gali paprasčiausiai keistis jų savininkai. Laiko atspindėjimo GIS sistemoje iššūkiai:

- Duomenų saugojimo arba duomenų struktūrų metodai, galintys apimti laike kintančius duomenis.
- Laiko informacijos užklausų teikimo metodai. Užklausų mechanizmai ne tik turi būti veiksmingi, jie turi apimti klausimus: „kada pokytis įvyko?“, „kokia kryptimi juda šis reiškinys?“ ir „kokia sparta vyksta pokytis?“ – klausimus, į kuriuos sunku atsakyti taikant dabartinius GIS metodus.
- Veiksmingo ir našaus laiko duomenų vaizdavimo metodai.
- Laiko interpoliavimo arba apibendrinimo metodai. Nors yra metodų, skirtų saugoti ir tirti kelis diskrečiuosius duomenų rodinius laiko atžvilgiu, priemonių, skirtų veiksmingai analizuoti duomenis tarp diskrečiųjų laiko rodinių, nėra.

Erdvinių objektų pokyčiams registruoti pasiūlyta daug duomenų modelių. Išanalizuosime kelias iš šių parinkčių, iliustruodami potencialius laiko duomenų valdymo sprendimus.

4.3.2 Nuotraukos modelis

Populiariausias laiko įtraukimo požiūris – tai **nuotraukos modelis**. Nuotraukos modelyje informacija laikoma žemėlapio sluoksnių seka, apibūdinančia tą patį reiškinį atskirais laiko momentais. Kiekviena nuotrauka arba „laiko pjūvis“ atspindi visą dominančią erdvinę apimtį ir paprasčiausiai apibūdina duotąją geografinių duomenų būseną duotuoju laiko momentu. Šis metodas buvo išvestas iš idėjos reguliariai fotografuoti tam tikrus dalykus. Toliau pateikiamame 4-19 pav. matote oro sąlygų dinamikos modelio nuotraukų seką.

4-19 pav. Nuotraukų požiūris į laiko modeliavimą

Šis modelis ypač gerai tinka darbui su rastriniais duomenų rinkiniais. Ši pateiktis išties būdinga nuotoliniams tyrimams, kur vaizdai gali būti gaunami tam tikrais intervalais, palydovui praskrendant virš konkrečios geografinės teritorijos. Jis taip pat gali būti

naudojamas su vektorinėmis pateiktimis ir abiem atvejais (rastro ar vektoriaus) šis metodas yra abstrakčiai paprastas ir nesunkiai įgyvendinamas.

Tačiau egzistuoja keli esminiai nuotraukų modelio trūkumai. Pirmiausia, kadangi kiekviena nuotrauka atspindi visą duomenų rinkinį kaip atskirą kopiją, egzistuoja žymus šio modelio taikymo pertekliškumas, ypač – ten, kur pokyčiai yra riboti. Be to, nėra loginio ryšio tarp atskirose nuotraukose saugomų geoobjektų. Pavyzdžiui, yra sunku atsakyti į klausimus apie žemės panaudojimo duomenų rinkinį, pvz.: „kuriose teritorijose buvo aktyviai dirbama žemė pastaruosius 5 ar 10 metų?“. Trečia, nuotraukų modelio prigimtis (diskrečiųjų stebėjimų įrašymas tam tikrais laiko momentais) reiškia, kad *tarp* fotografavimo momentų įvykę reiškiniai nefiksuojami. Mes negalime žinoti apie tikslų įvykių laiką – tik apie laikotarpį, kuriuo vyko konkretus pokytis.

Laiko tarpas tarp dviejų nuotraukų apibrėžia, kaip gerai mes suprantame pokyčio prigimtį. Kaip minėta pirmiau, negalime tiksliai žinoti, kada tarp fotografavimo momentų kas nors įvyksta. Mes galime tik matyti, kad kažkas matyti pirmoje nuotraukoje ir nematyti antroje. Kuo mažesnis laiko tarpas tarp dviejų gretimų nuotraukų, tuo tiksliau galime nustatyti, kada įvyko pokytis. Šis rodiklis vadinamas **laiko skaidymo gyliu** arba mažiausiu laiko intervalu, kurį galima atspindėti laiko GIS. Duomenų bazės laiko skaidymo gylis priklauso nuo specifinės GIS programos poreikių. Kai kurios programos (pvz., eismo srautų ar oro sąlygų stebėjimo) gali reikalauti ganėtinai tikslaus skaidymo gylio (minučių ar sekundžių), tuo tarpu bendruomenės planavimo programoms tarp dviejų gretimų nuotraukų gali prabėgti mėnesiai ar net metai.

Nuotraukas *ArcGIS* aplinkoje galima valdyti naudojant SDE versijas. Šiuo būdu galime išlaikyti mūsų duomenų nuotraukų seką laiko tėkmėje ir tuo pat metu palaikyti nuotraukų nelinijinį pobūdį. Pavyzdžiui, galima išanalizuoto kelis planavimo scenarijus, atspindinčius alternatyvias laiko atšakas, o ne nuotraukas išilgai laiko linijinio kelio. Kaip jau aptarėme pirmiau, versijų kontrolė nepasižymi kiekvienos versijos visos duomenų kopijos pertekliškumu, tad šio metodo saugojimo sąnaudos gali būti mažesnės. Tačiau daugelio geoobjektų klasės sudėtingų versijų išlaikymas skirtumų lentelėse gali brangiai kainuoti ir žymiai sulėtinti darbo spartą. Versijų kontrolė – geras sprendimas organizacijai, pageidaujančiai palaikyti ilgalaikę kintamos geoobjektų klasės istoriją, tačiau ji gali veiksmingai tikti ir palaikant nedidelį skaičių nuotraukų, kurių reikia greičiausiai tik trumpalaikiam planavimo procesui.

4.3.3 Regiono-Vieneto modelis

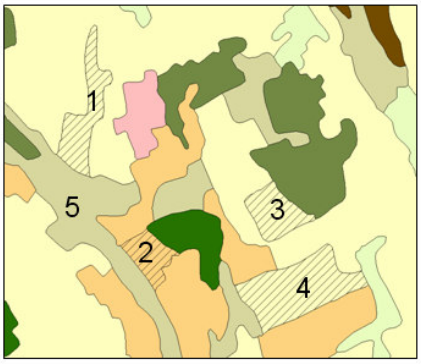
Kitas laiko įtraukimo į GIS būdas – žymėti kiekvieną kada nors egzistavusią eilutę ir taikyti tam tikrą mechanizmą (pvz., atributus) nurodyti duotojo objekto ar atributo vertės ribas konkrečiu laiko momentu. Šį požiūrį tekste paaiškinti yra gana sudėtinga, tačiau jis tampa ganėtinai aiškus pamačius rezultatus, tad norėdami iliustruoti, kaip šis modelis atspindi laiką, pasinaudosime pavyzdžiais.

Siekdami iliustruoti paprasčiausią šio modelio įdiegtį, apsvarstykime nedidelę Lietuvos CORINE žemės dangų duomenų rinkinio dalį. Šio duomenų rinkinio žemėlapiuose žemės panaudojimo mastelis siekia 1:10000, stebimi du laiko momentai: 1995 m. ir 2000 m. 4-20 pav. pateikiamas nedidelis šio duomenų rinkinio fragmentas, pritaikius nuotraukų požiūrį ir

siekiant parodyti 1995 – 2000 m. laikotarpio pokytį. Abiejuose vaizduose yra sunumeruoti plotai, kuriuose žemės panaudojimas 2000 m. lyginant su 1995 m. pakito. Visi sunumeruoti plotai 1995 m. buvo sukoduoti kaip dirbama žemė 231 (apleista ganykla, šviesiai ruda). 2000 m. 1, 3 ir 4 plotai buvo pakeisti į 211 klasę (nedrėkinama ariama žemė, šviesiai geltona), o 2 plotas pakito į 242 klasę (kompleksiniai žemdirbystes plotai, šviesiai oranžinė).

4-20 pav. CORINE žemės panaudojimas nuotraukų modelio požiūriu

Bendrojo pobūdžio nuotraukų modelio požiūris sprendžiant šio pokyčio uždavinį būtų palaikyti du žemės panaudojimo duomenų rinkinius: vienas jų būtų skirtas 1995 m., kitas – 2000 m. Abu būtų išsamūs žemės panaudojimo duomenų rinkiniai ir galėtų veikti atskirai. Regiono-vieneto modelio taikymo alternatyva – atspindėti abu laiko rodinius viename erdvinių duomenų rinkinyje ir panaudoti atributus apsprendžiant, kuri naudojama žemė priskirtina kuriai nuotraukai. 4-21 pav. matote vieną sprendimo būdą.



FeatureID	LU1995	LU2000
1	231	211
2	231	242
3	231	211
4	231	211
5	231	231
:	:	:

4-21 pav. CORINE žemės panaudojimo pokyčių Regiono-vieneto pateiktis

Pirmasis šios pateikties reikalavimas – tai, kad 1995 m. ir 2000 m. žemės panaudojimo daugiakampiai turi būti sulieti į vieną daugiakampių rinkinį, kuris funkciniu požiūriu juos kirstų. Žemėlapis kairėje rodo sulietus daugiakampius, kur pakeistos paskirties žemės plotai vaizduojami dryžuotu raštu. Daugiakampio spalvos vaizduojamos pagal 2000 m. spalvų gamą, tad studentai gali matyti, kaip 1995 m. buvusios apleistos ganyklos naudojamos 2000 m. Dešinėje pusėje pateikta lentelė rodo, kaip daugiakampio atributai atrodytų, įtraukus 1995 m. ir 2000 m. žemės panaudojimo atributus (jie saugomi – atitinkamai – *LU1995* ir *LU2000* laukuose).

Daugiakampiai, nuo 1995 m. iki 2000 m. nepakeitę klasifikacijos, 4-21 pav. vaizduojami kaip 5 daugiakampis, t. y. 1995 m. 2000 m. žemės panaudojimo vertės išlieka

nepakitusios. Daugiakampiai, pakeitę savo klasifikaciją, lentelėje pateikiami su *FeatureID* 1-4. Pavyzdžiui, 1 daugiakampis 1995 m. buvo užkoduotas kaip 231, o 2000 m. – kaip 211. Funkciniu požiūriu mes vis dar naudojame nuotraukų modelį, kadangi galime matyti žemės panaudojimą 1995 m. ir 2000 m., tačiau sušvelninome ženklią pertekliškumo problemą, integruodami abu daugiakampių rinkinius į vieną atributų lentelę. Šis metodas gali būti pritaikomas įtraukti atributus *FromDate* (nuo) ir *ToDate* (iki) ir sutalpinti daugiau laiko intervalų, o ne tik specifines, diskrečiąsias nuotraukas 1995 m. ir 2000 m.

Siekiant pažymėti arba užklausti žemės panaudojimo informacijos 1995 m., užtenka naudoti tik *LU1995* stulpelį. Siekiant pažymėti arba užklausti žemės panaudojimo informacijos 2000 m., užtenka naudoti tik *LU2000* stulpelį. Ši pateiktis supaprastina ir pokyčio statistikos skaičiavimus, kadangi atributų lentelė aiškiai nurodo tuos daugiakampius, kurių žemės naudojimo pobūdis pasikeitė.

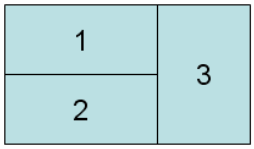
Vienas rimčiausių šios pateikties apribojimų – tai, kad jis nenurodo daugiakampių sankaupos. Pavyzdžiui, 1 daugiakampio regiono-vieneto pateiktis (4-21 pav.) nenurodo, kad šis daugiakampis 1995 m. buvo 5 daugiakampio dalis, tačiau 2000 m. tapo didesnio supančio geltono daugiakampio dalimi. Nors šis apribojimas nekelia problemų žemės panaudojimo taikymo sričiai (kai pagrindinis tikslas – paprasčiausiai suklasifikuoti duotąjį plotą), jis tampa itin svarbiu trūkumu nagrinėjant žemės nuosavybę. Pastarojoje pritaikymo srityje mums būtina suprasti, kad nuosavybės ir apmokestinimo požiūriais du arba daugiau diskrečiųjų daugiakampių – tai logiškai vienas žemės sklypas.

Siekdami išspręsti šią problemą, sudarysime papildomą lentelę: ji valdys aukštesnio lygio objektus, kurie paprastus fragmentuotus daugiakampius sujungs į logines sankaupas. Tai – tikrasis regiono-vieneto modeliavimas, pavadinimą gavęs iš dengimo duomenų struktūros *regiono* geoobjektų klasės (svarstytos šio kurso 2 dalis). Jame mes galime logiškai grupuoti paprastus geoobjektus bei formuoti sudėtingesnius.

Siekdami pailiustruoti šią sąvoką, peržvelgsime paprastą sklypų dalijimą, kaip parodyta 4-22 pav.

4-22 pav. Paprastas dalijimo pavyzdys

Šiame paprastame pavyzdyje kairėje esantis vaizdas rodo dviejų žemės sklypų laikotarpiu nuo 1980 iki 1992 m. 1993 m. pradžioje A sklypas padalintas ir sukurti du nauji sklypai: C ir D. Siekdami įtraukti šį pokytį, sujungsime šias dvi erdvines pateiktis ir visas įmanomas daugiakampių pateiktis įrašysime į vieną duomenų rinkinį. Visus gaunamus daugiakampius traktuosime kaip paprastus objektus arba primityvius daugiakampius. Pastarieji bus naudojami kaip statybinė medžiaga žemės sklypams, kurie bus atspindimi kaip regionai. Primityvieji daugiakampiai ir gaunama regionų-vienetų lentelė pateikiama 4-23 pav.



FeatureID	Polygon	FromDate	ToDate
A	1	1980	1992
A	2	1980	1992
B	3	1980	2007
C	1	1993	2007
D	2	1993	2007
:		:	:

4-23 pav. Dalijimo pavyzdžio regionų-vienetų pateiktis

Dešinėje esanti lentelė leidžia pateikti mūsų žemės sklypus (A, B, C ir D), naudojant primityvių daugiakampių 1, 2 ir 3 sankaupas. Pavyzdžiui, sklypas A iš pradžių susidėjo iš dviejų mažesnių primityvų – 1 ir 2. Siekiant tai atspindėti, į lentelę įtraukiami du įrašai, parodantys, kad A susideda iš šių dviejų primityvų. Kiekvieno įrašo lauko *ToDate* (iki) vertė yra 1992. Tai reiškia, kad jie nuo 1992 m. nebeegzistuoja. Nepakitę sklypai, pvz., B sklypas, paprasčiausiai turi vieną įrašą, nurodantį, kad jie egzistavo nuo 1980 iki 2007 m. Naujieji sklypai C ir D turi įrašus, kuriuose lauko *FromDate* (nuo) vertė yra 1993 – tai reiškia, jie buvo sukurti tuo metu.

Šio požiūrio problema – ta, kad gali būti sunku teikti užklausas ir vaizduoti. Esama GIS programų sąsaja šiai paskirčiai neturi pritaikytų funkcijų, tad siekiant užtikrinti, kad vartotojai galėtų peržiūrėti duotojo laikotarpio duomenų bazę, gali tekti sukurti sudėtingą apibrėžimo užklausų seką ir simboliką. Šis metodas taip pat generuoja gana žymią fragmentaciją ir dėl to jo valdymas yra brangus, kadangi reikia valdyti daug laiko pjūvių.

4.3.4 Kitos alternatyvos

Nors didžioji dauguma laiko įdiegčių grindžiamos nuotraukos ar regiono-vieneto modeliais arba tam tikromis jų variacijomis, egzistuoja keli alternatyvūs būdai GIS aplinkoje perteikti laiką. Keli iš jų reziumuojami toliau.

Atnaujinimo modelis saugo visą tik vieno duomenų rinkinio versiją, o įvykus pokyčiui nauja informacija įtraukiama atnaujinimų pavidalu. Atnaujinimuose nelaikoma duomenų bazės kopija: vietoj to įrašomi tik duomenų bazės pokyčiai. Tai atliekama būdu, panašiu į anksčiau aptartą operacijų žurnalo sąvoką.

Praktikoje šis metodas gali būti įgyvendintas naudojant *ArcMap* **archyvavimo** funkciją. Archyvavimas išnaudoja paprasčiausią mūsų aptartą laiko valdymo metodą: atributų *FromDate* (nuo) ir *ToDate* (iki) įtraukimą į geoobjektus, siekiant nurodyti laikotarpį, kuriuo galioja konkretus geoobjektas. Geoobjektų klasė, kuriai įjungta archyvavimo galimybė, kiekvieną kartą pakeitus geoobjektą sukurs naujus duomenų bazės vieneto (pvz., žemės sklypo) įrašus.

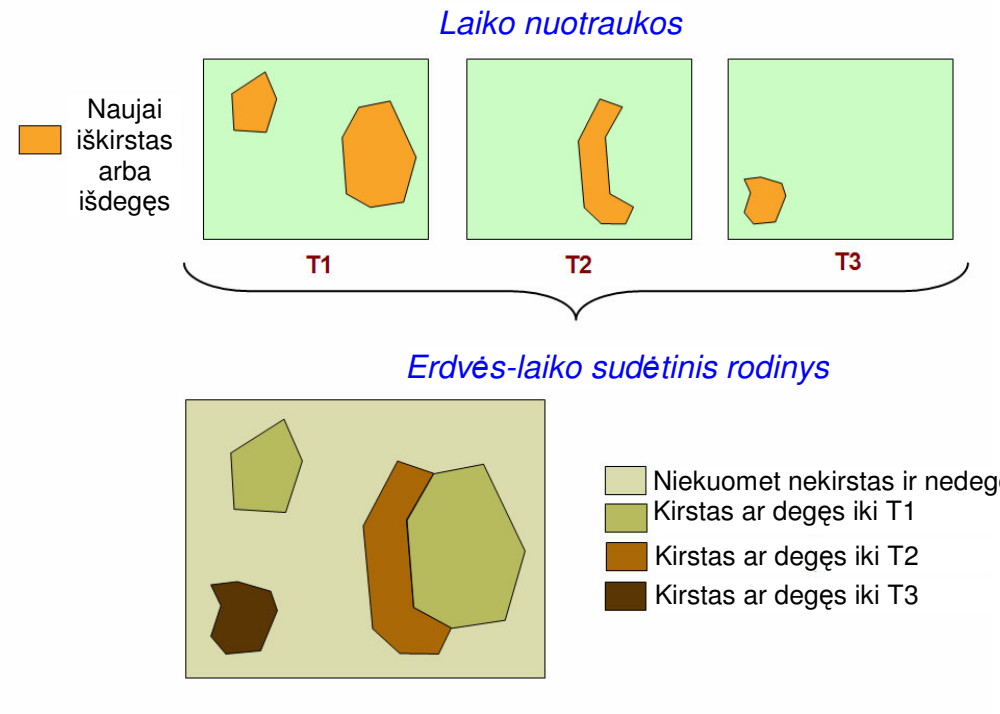
Pavyzdžiui, jei žemės sklypas šiandien buvo padalintas į du naujus sklypus, originalaus sklypo atributas *ToDate* (iki) nustatomas lygiu šiandienos datai – tai reiškia, kad nuo šiandienos šis sklypas nebegalioja. Taip pat sukuriami du nauji sklypų geoobjektai, kurių atributas *FromDate* (nuo) nustatomas lygiu šiandienos datai – tai reiškia, kad iki

šiandienos jie neegzistavo. Siekiant veiksmingai naudoti šiuos laiko įrašus, siūlomi keli programiniai įrankiai, kuriais naudojantis galima peržvelgti ir užklausti pokyčius laiko tėkmėje. Šis modelis pasižymi tam tikru pertekliškumu, kadangi dėl vieno tokio žemės sklypo padalijimo archyvinėse geoobjektų klasės lentelėse daromi trys įrašai, tačiau esminis skirtumas nuo nuotraukų modelio – tai, kad nepakeistas geoobjektas duomenų bazėje pasirodo tik vieną kartą.

Erdvės-laiko sudėtinis modelis yra panašus į regiono-vieneto modelį, kadangi praeities ir dabarties duomenis įrašo į tą patį sluoksnį. Čia laiko nuotraukos geografiniu požiūriu apdorojamos, įtraukiant visus pokyčius į vieną sluoksnį, kaip buvo daroma regiono-vietos metode. Tačiau tiksli duomenų bazės pateiktis bus skirtinga ir priklausys nuo programos. Rezultatu gali būti vienas atributas, nurodantis konkrečios nutikusios veiklos laiką arba laikas nuo to įvykio. Labai paprastu pavyzdžiu būtų galima iliustruoti vulkanų išsiveržimus. Paprastas išsiveržimų erdvės-laiko sudėtinis modelis gali saugoti taškines visų išsiveržimų vietas bei atributą, nurodantį, kaip seniai įvyko išsiveržimas. Tuomet galima naudoti simboliką arba užklausų klases ir keisti būdą, kuriuo rodysime arba analizuosime tokią geoobjekto klasę.

Šį požiūrį galima taikyti ir valdant miškus. 4-24 pav. pateikiamas erdvės-laiko sudėtinis modelis, vaizduojantis, kada vyksta miško kirtimo ar gaisro įvykis. Viršuje esantys vaizdai rodo kirtimo arba gaisro įvykius, nutikusius trimis laiko momentais. Sudėtinė versija (iliustracijos apačioje) rodo, kaip tai galima perteikti viena geoobjekto klase, galbūt – su atributu, informuojančiu apie laiko trukmę nuo miško kirtimo momento. Taikant šį metodą, galima apskaičiuoti kiekvienos miško zonos apytikslį amžių arba vizualiai suvokti, kaip miškas kito bėgant laikui.

Šis modelis gali būti veiksmingas dirbant su teminiu požiūriu paprastais duomenimis arba duomenimis, kuriuose yra nedaug atributų, turinčių nedaug verčių. Pavyzdžiui, 4-24 pav. funkciniu požiūriu perteikiamas dvejetainiu ryšiu: yra miškas arba jo nėra. Mėginant pavaizduoti pokyčius CORINE žemės panaudojimo srityje su įvairiomis žemės panaudojimo atributų vertėmis arba žemės nuosavybę su daugeliu žemės sklypų aprėpties pokyčių, šio metodo taikymas geriausiu atveju būtų keblus. Be to, keičiant duomenis, į sudėtinį sluoksnį būtina įtraukti nuotraukas ir reikia perskaičiuoti atributus.



4-24 pav. Erdvės-laiko sudėtinis modelis

ArcMap turi dar vieną mechanizmą, skirtą tvarkyti laiko duomenis. Tai – plėtinys pavadinimu **Tracking Analyst**. **Tracking analyst** yra pagrįstas sprendimas skirtais valdyti geoobjektus, kurie laikui bėgant juda arba keičia būseną. Tokių situacijų pavyzdžiai – judantys geoobjektai, pvz., transporto priemonės arba gyvūnai, arba stacionarūs objektai, keičiantys savo būseną, pvz., mašinų gamyba arba nuotolinis oro jutiklis. **Tracking analyst** gali dirbti su realaus laiko arba archyvuotais laiko duomenimis. Jis siūlo įrankių rinkinį, leidžiantį vartotojams peržvelgti ir užklausti geoobjektus laiko aspektu bei turi galimybę kurti animaciją, vaizduojančią judesius arba būsenos pokyčius.

4.4 Paskirstytosios DBVS

Kol kas didžiąja dalimi vengėme DBVS architektūros aptarimo. Nagrinėdami vienalaikiškumo klausimus, apsvarstėme daug pavyzdžių, kai keli vartotojai bando pasiekti tą pačią centrinę duomenų bazę (tai – kliento-serverio konfigūracijos pavyzdys). Tai – greičiausiai dažniausiais požiūris į prieigos prie duomenų suteikimą keliems vartotojams.

Paprasčiausias duomenų prieigos metodas, kai egzistuoja tik vienas GIS vartotojas – tai programą (pvz., *ArcMap*) ir geoduomenų bazę (pvz., failų geoduomenų bazė) įdiegti į vieną fizinį kompiuterį. Toks požiūris reiškia, kad nėra tinklo delsos arba konkurencijos tarp duomenų vartotojų, tad procesas yra itin veiksmingas. Tačiau kai vienoje organizacijoje yra daugiau nei vienas vartotojas, šis metodas daro duomenų bendro naudojimo darbo stotyse prielaidą, kuri iš tikrųjų nelabai tinka kažkam stambesniam nei nedidelis biuras.

Failų serverio požiūris apima failų bendrą naudojimą centriniame serveryje, pagal poreikį leidžiant šiuos failus pasiekti keliems vartotojams, kurie greičiausiai naudojasi į bendrą tinklą sujungtais kompiuteriais. Šis požiūris leidžia vartotojams bendrai naudotis duomenimis, tačiau jie negali tuo pačiu metu dirbti su duotuoju duomenų failu, o atsakomybė už saugą, atsargines kopijas ir atstatymą tenka patiems vartotojams. Šis modelis buvo plačiai paplitęs praėjusio dešimtmečio pradžioje, kai daugelis duomenų struktūrų buvo iš principo skirtos vienam vartotojui, pvz., *shape* failas.

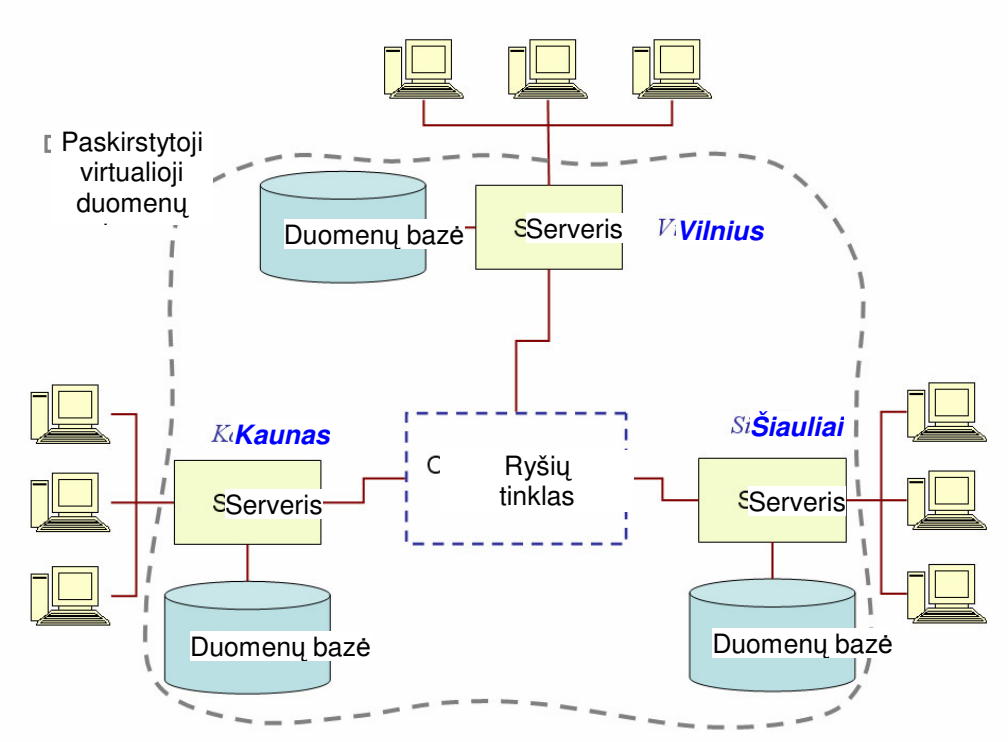
Dėl šių paprastesnių požiūrių į duomenų saugojimą ir prieigą prie jų apribojimų bei tinklų ir serverių technologijos pažangos kliento-serverio požiūris jau tapo tipiniu. Jis leidžia keliems vartotojams vienu metu pasiekti duomenis, kuriuos pateikia DBVS, kuri savo ruožtu gali kontroliuoti vienalaikiškumą, saugą, duomenų vientisumą, atsarginių kopijų darymą ir atstatymą. Tačiau kliento-serverio konfigūracija nėra vienintelis galimas būdas palengvinti duomenų prieigą. Šioje temoje nagrinėjamos kelios alternatyvios standartinio vieno serverio ir kelių klientų modelio konfigūracijos.

4.4.1 Paskirstytoji konfigūracija

Atliekant **paskirstytuosius skaičiavimus,** „atskiri kompiuteriai gali būti sujungiami į tam tikrą ryšių tinklą tokiu būdu, kad vieną duomenų apdorojimo užduotį būtų galima paskirstyti per kelis tinklo kompiuterius“ (Date, 2000, 50 p.). Sąvokos esmė – suskaidyti vieną stambią duomenų bazę į kelias mažesnes, lengviau valdomas duomenų bazes ir saugoti jas skirtinguose prie tinklo prijungtuose kompiuteriuose. Be to, apdorojimo procedūrą galima paskirstyti keliems atskiriems prijungtiems kompiuteriams. Susidomėjimas paskirstytuoju duomenų bazių valdymu išaugo dėl tinklo ryšio pažangos ir padidėjusios verslo operacijų sklaidos.

Paskirstytąją duomenų bazę (PDB) galima apibūdinti kaip „daugialypių logiškai susietų duomenų bazių, paskirstytų kompiuterio tinkle, rinkinį“, o **paskirstytųjų duomenų bazių valdymo sistemą** (PDBVS) – kaip „programinę sistemą, valdančią paskirstytąją duomenų bazę, išlaikant paskirstymo skaidrumą vartotojo atžvilgiu“ (Elmasri ir Navathe, 2000, 766 p.). PDBVS teoriškai pasižymi lankstumu ir galingumu, kurių neturi centralizuota sistema.

4-25 pav. pateikiamas paskirstytosios duomenų bazės konfigūracijos pavyzdys. Čia mes turime kelias geografiškai atskirtas vietas, kiekvienoje kurių veikia galiojančios atskiros duomenų bazių sistemos. Egzistuoja tam tikras diskrečiųjų fizinių duomenų bazių rinkinys, kuris, kartu funkcionuojant atskiriems kompiuteriams, tampa viena stambia *virtualia* duomenų baze. Kiekviena vietinė duomenų bazė saugo duomenis, labiausiai aktualius ir naudojamus vietos vartotojų, tačiau esant reikalui galima visuomet pasiekti nuotolinę duomenų bazę.



4-25 pav. Paskirstytosios duomenų bazės konfigūracija

Paskirstytosios duomenų bazės turi kelis pranašumus prieš įprastas centralizuotas duomenų bazes:

Didesnis pasiekiamumas	Duomenų bazės yra patikimesnės, nes mažiau remiamasi vienu centriniu kompiuteriu. Paskirstytoji sistema vieno kompiuterio gedimo atveju gali perleisti operacijas kitam kompiuteriui. Be to, kadangi duomenų bazės dalys yra laikomos atskirose vietose, mažiau tikėtina, kad kils užrakinimo ar nesutarimo problemų, kurios trikdytų prieigą.
Didesnė prieigos sparta	Kai stambi duomenų bazė yra paskirstoma per kelias vietas, kiekviena sudedamoji duomenų bazė būna mažesnė ir greičiau reaguoja į užklausas. Be to, kiekvienai smulkesnei duomenų bazei užklausas teiks mažesnis vartotojų skaičius nei būtų vienos stambios duomenų bazės atveju.
Plėtros palengvinimas	Paskirstytoje sistemoje lengviau įrengti papildomus procesorius ar padidinti duomenų bazės dydį.

Žinoma, šie patobulinimai turi savo kainą. Paskirstytosios duomenų bazės turi kelis trūkumus:

Padidėjęs sudėtingumas	Paskirstytąją duomenų bazę valdyti daug sudėtingiau nei centralizuotą. DBVS turi sutraukti duomenis iš kelių vietų, sinchronizuoti besidubliuojančius duomenis ir subalansuoti atskiriems mazgams tenkančias apdorojimo apkrovas. Be to, išauga procedūrų sudėtingumas, kadangi keliose atskirose vietose gali dirbti keli duomenų bazių administratoriai.
Padidėjęs saugyklos poreikis	Kai kuriems paskirstytiesiems modeliams (žr. toliau) reikia dubliuoti tam tikrus duomenis.

Egzistuoja du atskiri paskirstytosios duomenų bazės aspektai: apdorojimo paskirstymas ir duomenų saugojimo paskirstymas. **Paskirstytojo apdorojimo** principu duomenų bazės apdorojimo užduotys paskirstomos dviem arba keliems tinklo mazgams. Pavyzdžiui, pagal paskirstytąjį apdorojimą užklausų teikimo ir duomenų tvirtinimo procedūros gali būti vykdomos viename kompiuteryje, o rezultatų ataskaitų generavimas – kitame. **Paskirstytasis duomenų saugojimas** – tai įmonės duomenų saugojimas keliose logiškai susietose vietose. Siekiant tai atlikti, duomenys turi būti padalijami į tam tikrą skaičių duomenų bazės fragmentų, kurie yra paskirstomi dalyvaujančioms darbo vietoms. Priklausomai nuo konfigūracijos, šie fragmentai tam tikru lygiu gali dubliuotis (tačiau nebūtinai).

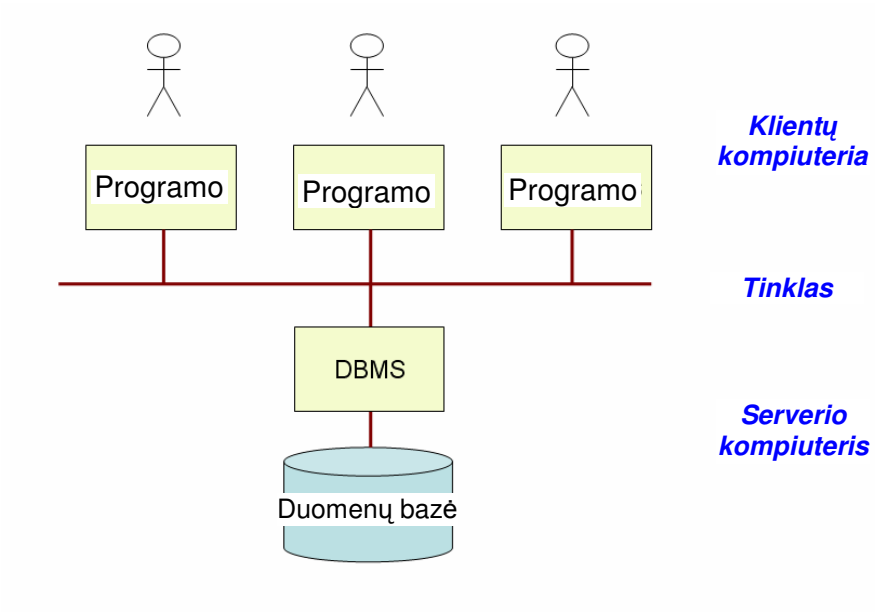
4.4.2 Kliento-serverio konfigūracija

Iš esmės beveik visos šiuolaikinės duomenų bazių sistemos sukurtos pagal kliento-serverio konfigūraciją. Tai – labai paprasta dviejų dalių struktūra, kai DBVS programa arba serveris valdo duomenų bazę, o tam tikras klientų programų skaičius šiam serveriui teikia užklausas. Klientų programomis gali būti įrankiai, pateikiami kartu su DBVS, skirti valdyti ar kitaip dirbti su duomenų baze; jomis taip pat gali būti su DBVS sąveikaujančios trečiųjų šalių programos. Kliento programos yra atsakingos už vartotojo sąsają ir apdorojimo veiksmus, būtinus atlikti prieš pateikiant rezultatus vartotojui. Serverio programa yra atsakinga už sąveiką su duomenų baze žemu lygiu, įskaitant duomenų bazės saugojimą ir jos prieigą. Žiūrint į pavyzdį, kuris bus pažįstamas GIS studentams, serveryje, kuriame laikomi mūsų geografiniai duomenys, gali veikti *SQL Server* SDBVS, o klientų programos būtų *ArcCatalog* arba *ArcMap*, kurios *SQL Server* teiktų duomenų užklausas ir vartotojui vaizduotų rezultatus. Visas duomenų saugojimo, vientisumo užtikrinimo ir prieigos kontrolės funkcijas atlieka *SQL Server*.

Ši konfigūracija skiriasi nuo failų serverio architektūros. Naudojant failų serverius, visas failas yra siunčiamas klientui ir kiekvienas klientas turi turėti visą DBVS programą, o duomenų vientisumas turi būti valdomas iš kliento programų. Pavyzdžiui, organizacija, centriniame failų serveryje (kurį gali pasiekti keli vartotojai) sauganti formų failų seką, geriau išnaudoja failų serverio konfigūraciją. Šioje situacijoje serveris paprasčiausiai pateikia formų failų turinį *ArcMap* klientams. *ArcMap* programa atsako už duomenų vientisumą. Priešingai – kliento-serverio duomenų bazėje yra remiamasi serverio

programine įranga, kuri apdoroja užklausas ir klientui nusiunčia tik reikalingą informaciją, taip žymiai sumažindama tinklo apkrovą.

Kliento ir serverio programos gali fiziškai būti laikomos tame pačiame arba atskiruose prie tinklo prijungtuose kompiuteriuose. Dažniausiai pasitaikantis kliento-serverio konfigūracijos tipas pateikiamas 4-26 pav. Čia keli kompiuteriai, kuriuose veikia klientų programos, bendrauja su vienu kompiuteriu, kuriame paleista serverio programa. Nors techniškai taip nėra visuomet, terminas „klientas-serveris“ čia taikomas beveik išskirtinai.



4-26 pav. Įprasta konfigūracija „vienas serveris – daug klientų“

Šio tipo konfigūracija – tai tam tikra paskirstytoji sistema, kadangi apdorojimo užduotis yra paskirstyta dviem kompiuteriams, tad šią architektūrą galime vertinti kaip specialų paskirstytosios sistemos pavyzdį. Kliento kompiuteryje gali būti apdorojami rezultatai ir pateikiami vartotojui, tuo tarpu serverio kompiuteryje pasiekiami duomenys ir perduodami rezultatai kliento programai. Tai – pagrindinis kliento-serverio architektūros pranašumas: yra galimybė atskirti apdorojimą ir taip padidinti našumą. Kitas patrauklus kliento-serverio konfigūracijos momentas – tai, kad šis kliento programų atskyrimas leidžia naudoti ne tokius galingus ir labiau paplitusius asmeninius kompiuterius. Šie kompiuteriai yra pigesni nei galingesni duomenų bazių serveriai ir personalo darbuotojai greičiausiai turės darbo su kompiuteriu įgūdžių.

Tačiau egzistuoja kelios kliento-serverio konfigūracijos naudojimo problemos, ypač – kai klientų ir serverių kompiuteriai yra fiziškai atskirti. Pirmiausia pasikliovimas tinklo jungiamumu reiškia, kad įtraukus daugiau vartotojų arba nuotoliniams vartotojams pasiekiant duomenų bazę mažėja sparta. Toks išdėstymas taip pat reiškia, kad gali būti patiriama didelių išlaidų, susijusių su stambaus centrinio duomenų bazių serverio palaikymu ir eksploatacija, o pasikliovimas šiuo centriniu kompiuteriu kelia patikimumo problemą.

4.4.3 Dauginimas

PDBVS veiksmingumas priklauso nuo jos gebėjimo paskirstyti duomenis tarp narių darbo vietų tokiu būdu, kuris palengvintų veiksmingą išgavimą. Vienas iš pagrindinių būdų suteikti vietinę prieigą prie paskirstytųjų duomenų – leisti daryti kopijas ir talpinti jas keliose fizinėse vietose. Tai yra naudinga našumo požiūriu, kadangi vietinė duomenų bazė nepatiria žymių tinklo laiko delsos problemų ir išvengiama vartotojų konkurencijos dėl duomenų. Tačiau paprasčiausias kopijų darymas ir jų platinimas kitiems personalo nariams ar biurams sukuria daug papildomo darbo, kadangi reikia sujungti kiekvienoje kopijoje atliktus pokyčius su dabartiniu duomenų bazės rodiniu. Be to, atsiranda klaidų tikimybė ir duomenų bazės nesuderinamumo galimybė.

Daugelis SDBVS programų turi galimybę kurti duomenų bazės ar tam tikros jos dalies dublikatus ir vėliau įtraukti atliktus pokyčius į kopiją. Kopijos yra vadinamos reprodukcijomis, o reprodukcijų kūrimo ir valdymo procesas – **dauginimu** (*replication*).

Dauginimas – tai dviejų etapų procesas. Pirmojo etapo metu iš pagrindinės duomenų bazės sukuriama viena arba kelios reprodukcijų duomenų bazės. Šios reprodukcijos gali būti paskirstomos ir naudojamos. Antruoju etapu, užbaigus taisymo darbus arba atsiradus ryšiui tarp reprodukcijų, šios atskiros versijos yra integruojamos. Šis procesas funkciniu požiūriu yra analogiškas versijos suderinimo procesui, kuris buvo aprašytas 3 dalyje, todėl kartais yra vadinamas **sinchronizacija**. Kalbant apie versijų suderinimą, problemas kelia tik tos situacijos, kai atliekami du to paties vieneto pakeitimai. Visi kiti pakeitimai gali būti integruojami automatizuotai.

Reprodukcijos yra patrauklios keliais aspektais. Keli iš jų reziumuojami toliau:

Prastas ryšys	Kai atskiri biurai yra atskirti prastu ryšiu arba ryšio išvis nėra, gali tekti naudoti reprodukcijas. Įprastas pavyzdys – kai personalo darbuotojai darbo metu turi vykti į nutolusias darbo vietas, pvz., su tikslu surinkti iš jų duomenis. Tokiu atveju į tą darbo vietą nešiojamame kompiuteryje vertėtų nusigabenti reprodukciją ir grįžus sinchronizuoti. Reprodukciją galima tam tikram laikotarpiui visiškai atjungti nuo tinklo.
Duomenų prieiga	Tais atvejais, kai vartotojai negali pasiekti duomenų dėl išteklių užsirakinimo, kopijavimas gali suteikti galimybę keliems vartotojams toliau dirbti, kol veiklos tinkle bus mažiau – tuomet galima atlikti sinchronizaciją.
Apkrovos balansavimas	Tuomet, kai vieni vartotojai itin intensyviai naudoja išteklius, kopijavimas gali leisti jiems išnaudoti vietos išteklius (pvz., savo kompiuteryje), o ne centrinės DBVS išteklius. Atskyrus ištekliams imlias užduotis į skirtingus kompiuterius, gali padidėti bendras sistemos našumas.
Kokybės užtikrinimas	Esant kokybės problemų, kopijavimo veiksmu galima izoliuoti visą duomenų bazę ar tam tikrą jos dalį. Užbaigus reprodukcijos apdorojimą, ją prieš integruojant į pagrindinę duomenų bazę galima nuodugniai peržvelgti.

Pavyzdžiui, rangovui arba konsultantui galima duoti reprodukciją ir ji prieš sinchronizaciją su pagrindine duomenų baze gali būti formaliai peržiūrėta ir patvirtinta.

4.4.4 Fragmentavimas

Be kopijavimo, yra dar vienas alternatyvus duomenų paskirstymo būdas, vadinamas **duomenų fragmentavimu**. Fragmentavimas – tai procesas, kurio metu atskiras duomenų objektas, pvz., duomenų bazės lentelė, yra suskaidomas į kelis fragmentus, kurie laikomi atskirose vietose. Kiekvienoje vietoje palaikomas **paskirstytųjų duomenų katalogas**, nustatantis kiekvieno fragmento vietą ir pobūdį – tai padeda apibrėžti, kaip būtinieji fragmentai bus surinkti, gavus užklausą ar esant reikalui atlikti operaciją. Egzistuoja trys pagrindinės duomenų lentelių skaidymo strategijos: horizontali, vertikali ir mišri fragmentacijos.

Siekiant iliustruoti šiuos fragmentavimo metodus, bus panaudota paprasta lentelė su informacija apie Lietuvos miestus. Jos atributai – tai miesto pavadinimas, apskritis, kuriai jis priklauso, gyventojų skaičius 2006 m. bei miesto centro X ir Y koordinatės. 4-27 pav. pateikiama ši lentelė.

CityID	Name	County	Pop2006	XCoord	YCoord
1	Alytus	Alytus	69,145	502600	6028900
2	Druskininkai	Alytus	16,641	492400	5986000
3	Vilnius	Vilnius	541,824	582600	6061500
4	Grigiskes	Vilnius	11,567	567000	6059200
5	Lentvaris	Vilnius	11,838	561600	6045800

4-27 pav. Pavyzdinė MIESTŲ lentelė

Vykdant horizontalų lentelių fragmentavimą, ji skaidoma į eilučių poaibius (fragmentus). Kiekvienas fragmentas yra laikomas atskiroje vietoje ir turi unikalias eilutes. Visi fragmentai pasižymi tais pačiais atributais. Horizontaliai fragmentuojant mūsų MIESTŲ lentelę, galbūt norėtume ją paskirstyti dviem apskritims, kiekviena kurių būtų atsakinga už savo teritorijoje esančius miestus. Tuomet logiška suskaidyti šią lentelę pagal tai, kuriai apskričiai priklauso konkretus miestas – tuomet kiekviename fragmente išliks visi atitinkamų miestų atributai. Tai – horizontalus fragmentavimas, kadangi į fragmentus skaidome eilutes. Gaunami fragmentai pateikiami 4-28 pav.

CityID	Name	County	Pop2006	XCoord	YCoord
1	Alytus	Alytus	69,145	502600	6028900
2	Druskininkai	Alytus	16,641	492400	5986000

CityID	Name	County	Pop2006	XCoord	YCoord
3	Vilnius	Vilnius	541,824	582600	6061500
4	Grigiskes	Vilnius	11,567	567000	6059200
5	Lentvaris	Vilnius	11,838	561600	6045800

4-28 pav. Horizontalus MIESTŲ lentelės fragmentavimas

Suskaidžius lentelę šiuo būdu, didžioji dalis darbų, toliau atliekama, pvz., Vilniuje, bus daroma su vietine duomenų baze. Jei vartotojui operacijai atlikti reikės visų Lietuvos miestų įrašų, PDBVS turės aptikti juos ir prieš pateikiant rezultatus atlikti fragmentų sujungimą. Taigi, PDBVS turi žinoti fragmentavimo kriterijus ir galėti iš nuotolinių fragmentų esant reikalui rekonstruoti stambesnius fragmentus arba visą lentelę. Šis procesas padaro PDBVS žymiai sudėtingesne.

Vertikalus lentelių fragmentavimas – tai jų skaidymas į atributų grupes (pagal stulpelius). Šis procesas yra panašus į normalizacijos procesą, kai atskiras „temas“ atitinkantys stulpeliai yra skaidomi į atskiras lenteles ir pagal poreikį užklausos teikimo atveju sujungiami. Tačiau čia lentelės skaidomos pagal verslo funkciją arba atliekant loginį grupavimą – pagal tai, kaip vartotojų grupės atskirose vietoje naudosis informacija.

Pavyzdžiui, galbūt mums reikia turėti skirtingas duomenų bazes dviejose vyriausybinėse agentūrose. Tarkime, kad vieną agentūrą domina miestų populiacija, kitą – tik jų geografinė vieta. Tokiu atveju mūsų MIESTŲ lentelę galima suskaidyti vertikaliai (žr. 4-29 pav.). Vėlgi – didžioji dalis užklausų greičiausiai ateis iš vietinio fragmento, tačiau jei bus poreikis gauti nuotolinius stulpelius, PDBVS turės atlikti sujungimo operaciją ir du stulpelių rinkinius sulieti į vieną bei tada pateikti duomenis.

CityID	Name	County	Pop2006
1	Alytus	Alytus	69,145
2	Druskininkai	Alytus	16,641
3	Vilnius	Vilnius	541,824
4	Grigiskes	Vilnius	11,567
5	Lentvaris	Vilnius	11,838

CityID	Name	XCoord	YCoord
1	Alytus	502600	6028900
2	Druskininkai	492400	5986000
3	Vilnius	582600	6061500
4	Grigiskes	567000	6059200
5	Lentvaris	561600	6045800

4-29 pav. Vertikalus MIESTŲ lentelės fragmentavimas

Mišraus fragmentavimo atveju sujungiamos horizontalaus ir vertikalaus fragmentavimo strategijos – taip, kad kiekviename fragmente būtų tam tikras eilučių ir stulpelių rinkinys. Taip veiksmingai galima suskaidyti mūsų MIESTŲ lentelę, jei abi apskritys turi du padalinius. Tokiu atveju galima būtų padalinti visą lentelę į keturis fragmentus (4-30 pav.).

CityID	Name	County	Pop2006
1	Alytus	Alytus	69,145
2	Druskininkai	Alytus	16,641

CityID	Name	XCoord	YCoord
1	Alytus	502600	6028900
2	Druskininkai	492400	5986000

CityID	Name	XCoord	YCoord
3	Vilnius	582600	6061500
4	Grigiskes	567000	6059200
5	Lentvaris	561600	6045800

CityID	Name	County	Pop2006
3	Vilnius	Vilnius	541,824
4	Grigiskes	Vilnius	11,567
5	Lentvaris	Vilnius	11,838

4-30 pav. Mišrus MIESTŲ lentelės fragmentavimas

Visose fragmentavimo metoduose atliekant pirminį užklausų apdorojimą bei jungiant duomenų bazes, patiriamos žymios pridėtinės išlaidos. Tačiau tikėtina, kad šias išlaidas kompensuoja faktas, jog daugeliu atvejų užklausos bus teikiamos tik vietiniams fragmentams ir taip greičiausiai sumažės reakcijos trukmė.

4.4.5 Paskirstytosios DBVS ir GIS

Keliose šio kurso vietose aptarėme, kaip GIS programa valdo geoduomenų bazės elgseną ir kaip duomenis valdo DBVS. Praeityje (pvz., su formų failais) beveik visa apdorojimo našta tekdavo grynai GIS programai ir duomenų bazė atlikdavo tik nedidelę dalį operacijų. Geoduomenų bazės modelis ir *ArcSDE* daug funkcijų perdavė DBVS, tad su išgavimu, atsarginėmis kopijomis ir atstatymų, sauga bei tam tikru duomenų vientisumo užtikrinimu susijusios operacijos nebepriklauso GIS (arba kliento) programai. Pastaruoju metu, SQL pritaikius standartinius geometrinius duomenų tipus (o ne dvejetainius koduotus BLOB laukus) ir erdvinius užklausų plėtinius, matome potencialą, kad pati DBVS atliks dar daugiau GIS anksčiau tekusio darbo.

Tačiau DBVS „nesupranta“ daugelio joje saugomų objektų reikšmės arba paskirties. Pavyzdžiui, DBVS (arba kitos trečiųjų šalių programos) versijuotų geoobjektų peržiūrai negali naudoti erdvinių SQL komandų, nes jos nesupranta PRADINĖS versijos ir pokyčio lentelių, kuriose laikomi duomenys apie pokyčius nuo PRADINĖS versijos sukūrimo. Šioms versijuotų geoobjektų klasėms peržvelgti mums tenka naudotis kliento programa, pvz., *ArcMap*.

Taip nutinka su paskirstytuoju apdorojimu ir duomenų saugojimu, kadangi jie yra susiję su GIS. Pavyzdžiui, beveik visos komercinės duomenų bazių programos palaiko kopijavimo galimybę, tačiau negalime naudoti DBVS kopijavimo mechanizmų, kadangi nutrauksime sudėtingų objektų – topologijos, ryšių klasių ir geometrinių tinklų – ryšius. Vietoj to reikia naudoti klientų programų siūlomus kopijavimo mechanizmus. Taigi, šiuo požiūriu galima naudotis tam tikrais PDBVS mechanizmais, tačiau negalime visiškai pereiti prie jų, kadangi didžioji manipuliavimo duomenimis dalis vykdoma kliento programoje.

Panaši situacija susidarė ir su paskirstytuoju apdorojimu bei paskirstytuoju duomenų saugojimu taikant fragmentavimą (vietoj kopijavimo). Kelios komercinės duomenų bazių sistemos, įskaitant *Ingres*, *Oracle* ir DB2, pradėjo judėti link **paskirstytojo skaičiavimo** sąvokos. Bazinė paskirstytojo skaičiavimo sąvoka – tai, kad visi tinklo ištekliai, įskaitant procesorius, kaupiklius, duomenų ir atminties įtaisus, turi būti sujungti į vieną loginį vienetą. Paskirstytojo skaičiavimo duomenų bazei teikiama apkrova balansuojama, pagal poreikį išnaudojant visus diskrečiuosius techninius ar programinius sistemos komponentus. Gaunama teoriškai greitesnė, pigesnė (kadangi nebereikia masyvių ir galingų duomenų bazių serverių) ir patikimesnė (kadangi išskirtinai nebesiremiame centriniu duomenų bazių serveriu) sistema. Tai – palyginti nauja technologija, su *Oracle* pirmąja paskirstytojo skaičiavimo versija (10g) išleista maždaug prieš metus.

Nors *ArcMap* palaiko *Oracle* paskirstytojo skaičiavimo versijas, iš jų gaunama ribotai naudos. Pamatinė DBVS, pvz., *Oracle*, negali efektyviai paskirstyti duomenų bazės fragmentų, jei ji negali tinkamai interpretuoti sudėtingų struktūrų lentelių (pvz., tų, kurios yra naudojamos versijų kūrimo procese, geometriniuose tinkluose bei topologijoje) turinio. Be to, kadangi vis dar didelė dalis operacijų atliekama GIS kliento programose, paskirstytojo skaičiavimo DBVS gebėjimas kontroliuoti apdorojimo galios paskirstymą yra ganėtinai ribotas.

Dalies klausimai savarankiškam darbui:

1. Kokie standartinės SQL elementai greičiausiai skirsis atskiruose SQL dialektuose?
2. Apibūdinkite skirtumus tarp SQL duomenų apibrėžimo komandų ir manipuliavimo duomenimis komandų.
3. Apibūdinkite našumo kompromisą, matricos indekse keičiant matricos dydį.
4. Kokie yra nuotraukų modelio apribojimai ir trūkumai, įtraukiant laiko matmenį į GIS duomenų bazę?
5. Apibūdinkite skirtumus tarp kopijavimo ir fragmentavimo kaip priemonių paskirstyti duomenų saugojimo funkcijas paskirstytojoje duomenų bazėje.

Literatūra

- Date, C.J. *An Introduction to Database Systems*. Addison-Wesley, 2000.
- Elmasri, R., and Navathe, S.B. *Fundamentals of Database Systems*. Addison-Wesley, 2000.
- ESRI. *Building a Geodatabase*. ESRI Press, 2005.
- ESRI. *ArcGIS On-line Help*. Viewed July, 2007.
- http://webhelp.esri.com/arcgisdesktop/9.2/index.cfm
- ESRI. *Working with the Geodatabase Using SQL*. ESRI Technical Paper, 2004. Viewed July, 2007.
- http http://downloads.esri.com/support/whitepapers/sde_/GeodatabaseUsingSQL_2_30mar06.pdf.
- McBride, S., Ma, D. and Escobar, F. *Management and Visualisation of Spatiotemporal information in GIS*. Presented at SIRC 2002 – The 14th Annual Colloquium of the Spatial Information Research Centre, University of Otago, Dunedin, New Zealand. Viewed July, 2007.
- http://www.business.otago.ac.nz/sirc/conferences/2002_SIRC/06_McBride.pdf
- Oracle Corp. *Grid Computing with Oracle*. An Oracle Technical White Paper. March, 2005. Viewed July 2007.
- http://www.oracle.com/technology/tech/grid/pdf/gridtechwhitepaper_0305.pdf
- Rob, P., and Coronel, C. *Database Systems: Design, Implementation and Management*. Thomson Learning, 2000.
- Shekhar, S., and Chawla, S. *Spatial Databases: A Tour*. Prentice-Hall, 2003.
- Zeiler, M. *Modeling Our World*. Redlands, CA: ESRI Press, 1999.