KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Justas Preibys

Duomenų vientisumo apribojimų realizavimo strategijos didelėje įmonėje tyrimas

Magistro darbas

Darbo vadovas
prof. L. Nemuraitė

Kaunas, 2008

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Justas Preibys

Duomenų vientisumo apribojimų realizavimo strategijos didelėje įmonėje tyrimas

Magistro darbas

Recenzentas
Doc. Dr. V. Kiauleikis
2008-01-

Vadovas
prof. L. Nemuraitė
2008-01-

Atliko
IFM-2/4 gr. stud.
Justas Preibys
2008-01-10

Kaunas, 2008

Investigation of strategy for implementation of data integrity constraints in large application system

Summary

Organizations, which operates with large amounts of data, often meets data integrity problems. The practice shows, that implementing data constraints only with methods of database management system (DBMS) slows database processes, loosing many time executing queries, and sometimes even crashes database work, because functional and reliable data control takes many system resources. Problems like than can be decreased, not loosing data integrity and quality, if a part of data integrity constraints are implemented in database, and other part – in application system, which communicates with it.

Effectiveness of constraints implementation is very relevant, when needs to be performed complicated financial, production management or scientific calculations. So the biggest attention needs to be dedicated to the methodology of implementation strategy for data integrity constraints when projecting, designing or maintaining databases.

This thesis describes types of data integrity constraints, the ways for implementation these constraints, and their analyzed advantages and disadvantages, also optimization for database queries. It is chosen most effective way for implementation every type of data integrity constraint, according to their run duration and complexity of implementation characteristics, and experimentally approved hypotheses, which were raised during analysis. With reference to the results of the experiment, is made and described the methodology of implementation strategy for data integrity constraints, which could help effectively implement data integrity constraints for large application systems, using as less as it is possible time, workforce and system resources, and at the same time ensuring data integrity and correctness.

Key words: data integrity, data constraint, trigger, database

Turinys

Paveikslėliai

Lentelės

1. Įvadas

Šiais informacinės visuomenės laikais pagrindinis žmonijos tikslas – kompiuterizuoti kuo įmanoma daugiau procesų, tarp jų ir verslo srityje. Stengiamasi diegti naujausias ir pažangiausias technologijas, kurios užtikrintų saugų, patikimą, spartų organizacijos vystymą, palengvintų fizinį ir protinį žmonių darbą. Dabar jau sunkiai įsivaizduojama nors ir mažos įmonės sėkminga ir produktyvi veikla be kompiuterizuotos informacinės sistemos. O ką jau kalbėt apie stambias organizacijas, kurios kiekvieną dieną operuoja milžiniškais kiekiais duomenų, kaupia juos, apdoroja, analizuoja, kas leidžia sėkmingai ir nuosekliai vystyti veiklą. Tokiem informacijos srautams dažniausiai reikalinga talpi ir patikima duomenų bazė. Įsivaizduokim, ką seniau reikėdavo kaupti dideliuose duomenų archyvuose, užimant nemažai vietos, rasti reikiamą informaciją tokioje gausoje duomenų, analizuoti ją paskendus šimtuose ar net tūkstančiuose lapų, dabar visą tai stambi organizacija gali sutalpinti vos spintos dydžio serveryje su duomenų baze.

Tokios technologijos leidžia sutaupyti nemažai laiko, vietos ir darbo resursų, kadangi didžiąją dalį procesų ir skaičiavimų valdo ir atlieka kompiuterizuota sistema. Tačiau, kad visą tai įgyvendinti ir sėkmingai plėtoti, reikalinga didelė ir itin griežta duomenų kontrolė, kadangi, kad ir kokia moderni ir funkcionali bebūtų naudojama technologija, lemiamą vaidmenį atlieka vartotojas. Būtent nuo sistemos vartotojo naudojamų duomenų sekų, kuriais vėliau operuoja sistema, priklauso jos veikimo patikimumas ir vientisumas.

Yra nemažai priemonių ir būdų užtikrinti korektiškų ir vientisų duomenų srautus duomenų bazėse. Iš praktikos pastebėta, kad duomenų apribojimus realizuojant tik duomenų bazės priemonėmis, sulėtėja jų darbas, prarandama nemažai laiko resursų vykdant užklausas, o kartais sutrinka ir pačios duomenų bazės veikimas. To pasėkoje yra nemaža tikimybė gauti netikslius, iškraipytus duomenis, o kartais netgi ir visiškai juos neatstatomai prarasti.

Apribojimų realizavimo efektyvumas ypatingai aktualus didelėse duomenų bazėse, kai reikia atlikti sudėtingus finansinius, gamybos valdymo, mokslinius skaičiavimus ir pan. Organizacijos, operuojančios dideliais duomenų kiekiais, dažnai susiduria su duomenų vientisumo problema. Tačiau norint užtikrinti operavimą korektiškais duomenimis, neprarasti jų, reikia sukurti sudėtingą mechanizmą, o taip pat nemažai investuoti į techninę įrangą, kadangi funkcionali ir patikima duomenų kontrolė reikalauja nemažų kompiuterinės sistemos resursų.

Tačiau kuriant ir realizuojant tokius sudėtingus duomenų vientisumo apribojimų mechanizmus, labai svarbu įvertinti galimą situaciją, atlikti eksperimentus su įvairiomis apribojimų kombinacijomis, optimizuoti juos, nes didelis kiekis apribojimų ir taisyklių ne tik padidina duomenų vientisumo saugumą, tačiau ir klaidingo sistemos veikimo tikimybę, metodų prieštaringumą. Tokias problemas didelių įmonių duomenų saugyklose įmanoma sumažinti,

neprarandant duomenų kokybės, dalį duomenų vientisumo apribojimų realizuojant duomenų bazėje, o kitus – su ja tiesiogiai komunikuojančioje taikomojoje programoje.

Todėl šio darbo tyrimo sritis yra didelių įmonių duomenų bazių su vientisumo apribojimais projektavimas, kūrimas ir palaikymas, o tyrimo objektas – duomenų vientisumo apribojimų įgyvendinimo strategijos pasirinkimo procesas.

Šio darbo tikslas – padidinti didelėse duomenų bazėse saugomų duomenų vientisumo apribojimų užtikrinimo efektyvumą, sukuriant vientisumo apribojimų įgyvendinimo metodiką.

Norint pasiekti tokį tikslą darbe reikės atlikti tokius uždavinius:

- Išanalizuoti duomenų bazių valdymo sistemos duomenų vientisumo apribojimų tipus ir jų realizavimo principus.
- Ištirti apribojimų realizavimo DB ir taikomosiomis programomis privalumus bei trūkumus įvairių apribojimų tipų atvejais.
- Sudaryti vientisumo apribojimų realizavimo rekomendacijas ir metodiką.
- Pritaikyti ją konkrečiai informacinei sistemai.

Analizuojant duomenų bazių valdymo sistemos duomenų vientisumo apribojimų tipus bei jų realizavimo būdus buvo remtasi literatūros šaltiniais [1], [2], [4] ir [11], kuriuose aprašomi duomenų bazių struktūriniai elementai, duomenų apribojimų kūrimas ir realizavimas kuriant duomenų bazę, jų silpnosios bei stipriosios vietos. Trigerių detaliai atalizei, jų veikimo principų, taikymo sričių ir atvejų analizei pasirinkti [6], [7], [8], [9] ir [10] literatūros šaltiniai. Tačiau juose pateikiama gana objektyvi nuomonė apie trigerius, atskleidžiamos jų galimybės, struktūra, veikimas, bet nėra konkrečių kriterijų, pagal kuriuos galima būtų spręsti, kada pravartu naudoti šį duomenų vientisumo apribojimų realizavimo metodą, o kada verčiau pasirinkti alternatyvų. Dauguma analizės metu išskirtų prioritetų bei teigiamų ir neigiamų apribojimų realizacijos principų buvo parinkti remiantis asmenine patirtimi ir darbo praktika.

Taigi darbo metu išanalizuoti išanalizuoti šie duomenų vientisumo apribojimų tipai:

- pirminio rakto;
- išorinio rakto;
- būtinos reikšmės (NOT NULL);
- reikšmės apibrėžimo;
- vidinio unikalumo;
- išorinio unikalumo;
- kardinalumo;

- išvestinės reikšmės.

O taip pat išanalizuoti duomenų vientisumo apribojimų būdai:

- Duomenų bazės apribojimų būdai:
  - Pirminis raktas
  - Išorinis raktas
  - CHECK funkcija
  - Trigeris
- Taikomoji programa (apribojimų valdiklis)

Atlikus skirtingų apribojimų tipų realizavimo galimybių analizę, prieita preliminarių išvadų, kaip geriau juos realizuoti – taikomojoje programoje ar duomenų bazėje. Tokie apribojimų tipai, kurie gali būti tiesiogiai realizuojami duomenų bazės valdymo sistemos priemonėmis, t.y. pirminio ir išorinio rakto, arba vidinio unikalumo, jomis turėtų ir būti realizuojami, kadangi šios priemonės veikia kur kas sparčiau nei tokį tikrinimą atliekant iš apribojimų valdiklio, o taip pat jos pagal prioritetą suveikia pirmiau, nei kitas duomenų bazės apribojimų mechanizmas – trigeris. Tačiau kiti duomenų apribojimų tipai – būtinos rekšmės ir reikšmės apibrėžimo, kurie taip pat gali būti realizuoti duomenų bazės valdymo sistemoje atitinkamai parametru NOT NULL ir funkcija CHECK, turėtų būti įtraukti į apribojimų valdiklį, kadangi šiems apribojimams neturi įtakos kiti lentelės įrašai, juos patikrinti galima taikomojoje programoje nesiunčiant užklausos į duomenų bazę.

Kiti, sudėtingesni duomenų vientisumo apribojimų tipai, tokie kaip išorinio unikalumo, kardinalumo, išvestinės reikšmės, ar refleksyvinių ryšių, kurių realizacijai nepakanka integruotų DBVS priemonių, reikalingas papildomas programavimas ir adaptacija, gali būti kuriami trigerių arba apribojimų valdiklio pagrindu. Išorinio unikalumo ir kardinalumo apribojimai sparčiau veikia juos realizavus apribojimų valdiklyje, kadangi visas duomenų vientisumo tikrinimas atliekamas taikomojoje programoje ir, jeigu tenkinamos sąlygos, tik tada siunčiamas įrašas įterpimui, priešingu atveju įrašo siuntimas į duomenų bazę net nevykdomas. Išvestinės reikšmės apribojimas nevykdo jokių tikrinimo sąlygų, o apskaičiuota reikšmė visuomet siunčiama duomenų bazei, todėl patogiau ir praktiškiau yra visas operacijas vykdyti viename apribojimų mechanizme – šiuo atveju trigeryje. Pastaruoju būdu turėtų būti realizuojamas ir refleksyvinių ryšių apribojimas, kuris naudojamas norint užtikrinti tam tikrų įrašų seką ar kitokį jų kombinacijų neprieštaringumą, kadangi apribojimų valdiklyje jį realizuoti būtų ganėtinai sunku, dėl įvairių išorinių raktų bei ciklišKumų tikrinimo.

Analizės metu sudarytų preliminarių hipotezių patikrinimui didelėje įmonėje sukurta eksperimentinė gamybinės sistemos duomenų bazė bei atliktas eksperimentas, kuris patvirtino, kad hipotezės buvo teisingos, ir leido nustatyti statistines apribojimų užtikrinimo priemonių veikimo charakteristikas. Eksperimento metu ištirtos išorinio unikalumo, kardinalumo, išvestinės reikšmės ir kombinuotų apribojimų veikimo charakteristikos, realizuojant juos skirtingais duomenų vientisumo apribojimų realizavimo būdais. Tyrimo rezultatai leidžia sudaryti apribojimų realizavimo strategiją bei jos parinkimo metodiką.

Analizės išvados bei eksperimento metu gauti rezultatai leidžia suformuluoti rekomendacinius apribojimų realizavimo strategijos parinkimo metodikos principus projektuojant, kuriant ir palaikant duomenų bazę:

- Sudaryti duomenų bazės schemą su vientisumo apribojimais.
- Pirminio, išorinio rakto, vidinio unikalumo apribojimus realizuoti DB.
- Išorinio unikalumo ir kardinalumo apribojimus realizuoti valdikliu.
- Išvestinės reikšmės arba dideles programines operacijas atliekantiems apribojimams atlikti eksperimentą.
- Jei yra daug apribojimų ir įvairių jų kombinacijų, tikslinga eksperimento būdu nustatyti, kokiomis priemonėmis ir konkrečiomis realizacijomis įgyvendinti tokius apribojimus.

Darbo struktūra:

- Analizės dalyje atlikta duomenų bazių valdymo sistemos duomenų vientisumo apribojimų tipų ir jų realizavimo būdų analizė, apibrėžti veikimo principai, panaudojimo galimybės, bei įvertinti privalumai ir trūkumai.
- Trečiame skyriuje pateiktas galimas sistemos modelis su duomenų vientisumo apribojimų realizavimo metodika. Atskirų apribojimų tipų realizavimui parinkti skirtingi būdai, įvertinus jų veikimo kokybės charakteristikas atskirais atvejais.
- Eksperimento dalyje aprašyta tyrimo eiga, pagal parinktos eksperimentinės sistemos modelį. Pateiktos išorinio unikalumo, kardinalumo, išvestinės reikšmės bei kombinuotų apribojimų išmatuotos veikimo trukmės bei apskaičiuoti parametrai, juos realizuojant skirtingais būdais.
- Penktame skyriuje sudaryta duomenų vientisumo apribojimų realizacijos strategijos parinkimo metodika, pateiktos rekomendacijos kuriant didelias duomenų bazes.
- Šeštame skyriuje suformuluotos darbo išvados.

Darbo tema parengtas straipsnis, kuris įteiktas konferencijai „IT2008“.

2. Duomenų vientisumo apribojimų realizavimo metodų analizė

Darbo tikslas – padidinti didelėse duomenų bazėse saugomų duomenų vientisumo apribojimų užtikrinimo efektyvumą, sukuriant vientisumo apribojimų įgyvendinimo metodiką.

Šiam tikslui pasiekti reikalinga išanalizuoti duomenų apribojimų tipus bei įvertinti jų realizavimo būdus, priklausomai nuo apribojimo tipo, duomenų įrašų kiekio, kreipinių kiekio ar dažnumo. Nustatyti trigerių ir procedūrinių apribojimų privalumus ir trūkumus, jų panaudojimo atvejus, kad būtų užtikrinta reliacinių duomenų darna šiuolaikinėse duomenų bazių sistemose. Išanalizuoti duomenų bazių valdymo sistemų (MS SQL Server, Oracle) duomenų vientisumo apribojimų principus, palyginti juos. Rasti optimalias užklausų struktūras.

Viena iš pagrindinių problemų, su kuriomis susiduria organizacijos, naudojančios kompiuterizuotas informacines sistemas – duomenų vientisumas ir korektiškumas. Kad duomenų bazė užtikrintų vientisumą, ji turi tenkinti tam tikrus apribojimus ir sąlygas duomenims bei išsaugoti ir pritaikyti tas sąlygas keičiant, šalinant ar įterpiant duomenis. Taip pat duomenų bazė turi vienodai reaguoti į įvairių vartotjų planuotas ir neplanuotas užklausas. Taigi bus tiriami duomenų bazių valdymo sistemų (DBVS) duomenų apribojimų mechanizmai ir jų funkcionalumas. Tačiau tokios duomenų kontrolės reikalauja nemažai sistemos resursų, kas įtakoja sistemos stabilumą ir veikimo spartą.

Pagrindiniai siekiamos sistemos vartotojai – visi įmonės ar kitos organizacijos duomenų baze disponuojantys asmenys. Tai gali būti tiek patyrę IT srities specialistai, tiek labai mažai kompiuterinių žinių turintys vartotojai, tačiau visi jie tiesiogiai susiję su operavimu duomenimis duomenų bazėje.

2.1. Vientisumo apribojimų tipai

Sėkmingam duomenų bazės realizavimui reikalinga griežta duomenų kontrolė, taigi DBVS gamintojai yra numatę ir sukūrę nemažai priemonių duomenų vientisumui užtikrinti. Išanalizavus pagrindinius duomenų apribojimus, buvo suformuluoti ir įvertinti kiekvieno jų privalumai ir trūkumai. Sunku nustatyti, kuris iš jų yra tinkamiausias ir optimalus, pradžioje tik išsiaiškinkime, kokie duomenų kontrolės principai tinka atskirai situacijai, o bendras jų rinkinys – užtikrina pakankamai aukšto lygio duomenų vientisumą ir darną duomenų bazėje.

Griežčiausias duomenų vientisumo apribojimas, kurį visapusiškai DBVS naudoja duomenų darnai užtikrinti, yra lentelės laukų taisyklės, kuriomis nurodomi konkretūs kiekvieno lauko duomenų tipai, simbolių kiekiai, reikšmės intervalai, šrifto koduotė ar netgi reikšmes pagal nutylėjimą. Taip pat kontroliuojama ar laukas gali turėti tuščią (NULL) reikšmę, ar ne. Toks

apribojimas tikrina fizinę duomenų struktūrą, t.y. pagal jų sudarymo vienetus (skaičius, raides, jų kombinacijas), simbolių kiekį, įvertina reikšmės tinkamumą lentelės laukui. Tai yra privaloma duomenų kontrolė duomenų bazėje ir jos privalumas nesunkiai pastebimas – be tokios kontrolinės priemonės lentelėse turėtume didelę betvarkę, DBVS ir taikomosioms programoms būtų sudėtinga ar net neįmanoma nuskaityti, inicijuoti ar apdoroti duomenis, atlikti skaičiavimus, kadangi nuskaičius konkrečią reikšmę, programa nesugebėtų įvertinti ar tai kažkokia skaitinė vertė, ar tai simbolinis pavadinimas, aprašas, ar tai galbūt atitinkamu datos formatu aprašyta data ir laikas. Tokią duomenų kontrolę įgalina atlikti ne tik nurodomų duomenų tipas, bet ir simbolinis kiekis.

Simbolių kiekių kontrolė leidžia optimaliai išnaudoti serverio talpą ir sukonkretinti duomenų ilgį. Tarkim žinodami, kad asmens kodas susideda iš 11 simbolių, rezervuojame lentelės eilutėje tam laukui tokį kiekį simbolių, kiek būtina, kitus atlaisvindami, taip pat užtikrindami, kad per klaidą nebus įvesti papildomi simboliai, kurie neturėtų priklausyti šiam atributui. Tačiau reikalinga ir atvirkštinė kontrolė, kuri neleistų saugoti įrašus ne tik su pertekliniu simbolių skaičiumi, bet ir su tuščia (NULL) reikšme. Tokio apribojimo tikslas yra neleisti patekti į lentelę nepilniems įrašams, kurių trūkstamos reikšmės yra griežtai reikalingos vėlesnėms operacijoms su duomenimis. Tarkime lentelėje talpinami duomenys apie įmonės juridinius klientus, reiškias laukai, kuriuose saugomas kliento įmonės kodas ir kontaktinė informacija yra privalomi, t.y. jie negali būti tušti, kadangi prireikus susisiekti su įmone ar išrašyti sąskaitą, neturėsime būtiniausios informacijos. Akivaizdžiai matosi, kad tokia duomenų kontrolė padėtų išvengti šios situacijos ir saugoti tik tuos įrašus, kurie turi būtiniausių laukų reikšmes.

Kitas efektyvus apribojimas – galimos reikšmės intervalas. Kuriant lentelės laukus, iš karto galima apriboti duomenų reikšmių intervalus. Tai patogu tokiais atvejais, kaip pavyzdžiui turimas lentelės laukas ir jame planuojami saugoti prekių kiekiai. Kadangi kiekis visada yra teigiamas skaičius, tai prie lentelės lauko galima nurodyti CHECK Kiekis >= 0 apribojimą. Tai gana paprasta, bet patogi apribojimo priemonė.

Galima pabrėžti, kad lauko reikšmės tipo ir ilgio duomenų apribojimai yra privalomi duomenų bazei (jie realizuoti kiekvienoje DBVS) ir jie neįtakoja užklausų apdorojimo spartos, serverio apkrautumo, vienodai reaguoja į vartotojų užklausas. Pagrindinis jos tikslas - užtikrinti, kad duomenų bazėje būtų pilni, reikiami ir apibrėžto tipo duomenys.

Tačiau tokia laukų duomenų reikšmių fizinė kontrolė yra gana konkreti, beveik nepritaikoma išskirtinėms situacijoms ir jos anaiptol neužtenka vientisoms duomenų bazėms, kadangi tokiu būdu neįmanoma įvertinti logines sąlygas, pagal kurias sistema nustatytų reikšmės tinkamumą, ar ji tenkina tam tikras verslo sąlygas, ar nesukelia kritinės situacijos sistemoje, ar nepažeidžia logikos lyginant su kitais įrašais lentelėje ir visoje duomenų bazėje, ar nesidubliuoja įrašai lentelėje.

Pastarosios problemos sprendimui duomenų bazėse naudojamas pirminio rakto (angl. *primary key*) duomenų apribojimas. Tai yra indeksuotas lentelės laukas ar laukai, kurie yra esminiai ir kurių reikšmių kombinacija negali kartotis lentelėje. Šis apribojimas nėra privalomas lentelei, tačiau jį patogu naudoti, kai norima saugoti informaciją apie konkrečius objektus, o prireikus greitai ir nesunkiai identifikuoti įrašą ir išgauti reikiamą informaciją. Pagrindiniai pirminio rakto privalumai – pasikartojančių reikšmių kontrolė bei greita duomenų paieška lentelėje. Priešingai nei duomenų reikšmių fizinė kontrolė, kuri neturi įtakos užklausų vykdymo spartai, pirminio rakto įrašai yra indeksuoti, t.y. turi surūšiuotą aibę nuorodų į lentelės eilutes, kas ženkliai paspartina duomenų paiešką. Žinoma mažoms arba statistinėms lentelėms, kur konkretus įrašas neturi didelės reikšmės, o imamas tam tikras įrašų intervalas, toks apribojimas nebūtinas, netgi nepatartinas, kadangi pasikartojus tai pačiai reikšmei yra tikimybė prarasti įrašą, nes jis nebus įterptas. Tokiu atveju galima įvesti valdomus duomenų apribojimus, su konkrečiomis tinkamumo ir logiškumo sąlygomis.

Kitas panašus duomenų vientisumo apribojimo tipas - atributo unikalumo identifikatorius, kuris naudojamas lentelės lauko unikalumui nusakyti. Jis kaip ir pirminio rakto apribojimas neleidžia lentelėje atsirasti besidubliuojančioms to lauko reikšmėms. Šį apribojimą galima sukurti lentelės kūrimo arba apribojimų papildymo metu.

Dažnai svarbu užtikrinti duomenų vientisumą ne tik vienos, bet kelių lentelių atžvilgiu. Vienas tokių apribojimų – išorinio rakto (angl. *foreign key*). Šis apribojimas yra naudojamas norint susieti lenteles tarpusavio ryšiu. Jis kuriamas lentelės laukui, kuris turi būti susietas su kitos lentelės pirminiu lauku. Toks apribojimas labai praverčia, kai norima laikytis tam tikros įvykių sekos, pavyzdžiui į lentelę negali būtį įvesta reikšmė, kol ji nesukurta kitoje lentelėje, arba apsaugant duomenų ryšius pašalinant įrašą, tuomet pašalinimas nebus vykdomas, jeigu ši reikšmė egzistuoja kitoje lentelėje, kuri siejasi per nurodomąjį atributo apribojimą.

Visgi išorinio duomenų unikalumo apribojimo duomenų bazės valdymo sistemose nėra numatyta. Jis gali būti atskirai programuojamas trigerių arba procedūrinių apribojimų pagalba. Toks apribojimas naudojamas, kai reikia identifikuoti keliose lentelėse susietus egzistuojančius atributus. T.y. tarkime turime lentelę, kurioje tam tikro lauko reikšmės gali kartotis, tačiau šios reikšmės kombinacija su kitos lentelės susieto lauko reikšme privalo būti unikali. Tuomet programuojamas apribojimas, kuris tikrintų ar nėra pažeidžiamas toks duomenų vientisumas.

Taip pat gali būti programuojami ir kiti duomenų vientisumo apribojimai: išvestinio atributo, kardinalumo ir refleksyvių ryšių. Primojo apribojimo, išvestinio atributo, principas yra nesudėtingas - lauko reikšmė gaunama pagal kitų laukų reikšmių apskaičiavimus ar reikšmių apdorojimus. Tokie apribojimai dažniausiai naudojami, kai pagal tam tikras įvestas reikšmes išvedami įvairūs finansiniai, moksliniai ar gamybiniai skaičiavimai. Kardinalumo apribojimas kiek sudėtingesnis. Jo

esmė yra apriboti lentelės įrašų kiekį, susijusių su kitos susietos lentelės vienu įrašu. Dažniausiai tai būna išoriniu raktu susietų laukų įrašų tikrinimas.

Dar vienas programuojamas duomenų vientisumo apribojimas – refleksyvių ryšių apribojimas. Tokia kontrolė reikalinga, norint užtikrinti tam tikrų įrašų seką ar kitokį jų kombinacijų neprieštaringumą. Apribojimas gali būti dviejų tipų – asimetrinis (angl. *asymmetric*), kuris neleidžia atsirasti įrašams su priešinga veiksmų seką, t.y. jeigu veiksmų kryptis yra 1 → 2, tai turėtų neleisti įterpti įrašą su seka 2 → 1, ir antisimetrinį (angl. *antysymmetric*) apribojimą, kurio veikimo principas panašus kaip ir asimetrinio apribojimo, tik jis leidžia tai pačiai reikšmei dalyvauti abejose apribojimo rolėse.

1 lentelė. Apribojimų tipai

Apribojimo tipas	**Realizuojamas DB**	**Realizuojamas apribojimų valdiklyje**
Pirminis raktas	Apribojimas integruotas DBVS ir veikia sparčiau nei apribojimų valdiklyje. Priskiriamas lentelės kūrimo arba koregavimo metu	Prieš įterpinėjant įrašą galima patikrinti ar jau nėra įterptos tokios reikšmės lentelėje
Išorinis raktas	Apribojimas integruotas DBVS ir veikia sparčiau nei apribojimų valdiklyje. Priskiriamas lentelės kūrimo arba koregavimo metu	Prieš įterpinėjant įrašą galima patikrinti ar tokia reikšmė egzistuoja kitoje lentelėje
Vidinio unikalumo	Apribojimas integruotas DBVS ir veikia sparčiau nei apribojimų valdiklyje. Priskiriamas laukui jo kūrimo arba koregavimo metu sakiniu UNIQUE	Prieš įterpinėjant įrašą galima patikrinti ar reikšmė bus unikali lentelės lauke
Būtinos reikšmės	Apribojimas integruotas DBVS, priskiriamas laukui jo kūrimo arba koregavimo metu sakiniu NOT NULL	Šiam apribojimui neturi įtakos kiti lentelės įrašai, juos patikrinti galima apribojimų valdiklyje nesiunčiant užklausos į duomenų bazę
Reikšmės apibrėžimo	Apribojimas integruotas DBVS, priskiriamas funkcija CHECK	Šiam apribojimui neturi įtakos kiti lentelės įrašai, juos patikrinti galima apribojimų valdiklyje nesiunčiant užklausos į duomenų bazę
Išorinio unikalumo	Galima realizuoti trigeriu	Galima patikrinti įrašų unikalumą susietose lentelėse prieš įterpinėjant įrašą. Esant nekorektiškam įrašui, užklausa į DB net nesiunčiama

Kardinalumo	Galima realizuoti trigeriu	Galima patikrinti kardinalumą lentelėse prieš įterpinėjant įrašą. Esant nekorektiškam įrašui, užklausa į DB net nesiunčiama
Išvestinis atributas	Galima realizuoti trigeriu	Apribojimas neturi tikrinimo sąlygų, apskaičiuotą reikšmę visais atvejais įterpia į lentelę, taigi visos užklausos siunčiamos į DB
Refleksyvių ryšių	Realizuojamas trigeriu, kadangi kitaip sunku būtų kontroliuoti įrašų seką pagal išorinius ranktus	Apribojimą realizuoti labai sudėtinga

2.2. Vientisumo apribojimų realizavimo būdai

Išanalizuotoje literatūroje nemažai šnekama apie duomenų kontrolės būdus, jų privalumus ir trūkumus, tačiau konkretaus atsakymo, kuri metodika būtų efektyviausia, sutikti netenka. R. Barono knygoje „Duomenų Bazių Valdymo Sistemos“ [2] smulkiai aprašomas ir rekomenduojamas trigerių mechanizmas, tuo tarpu Morgan Kaufmann leidinyje „SQL Practical Guide for Developers“ [1] patariama kuo mažiau ir atsargiau naudoti šią priemonę. Kitose knygose ir leidiniuose ši problema menkai aptariama, kalbama labai apibendrintai. Išvados ir sprendimas suformuluotas remiantis asmenine darbo su duomenų bazėmis ir duomenų vientisumo apribojimais jose patirtimi, išanalizavus veikimą ir funkcionalumą.

Nemažai informacijos apie DBVS skirtumus aptinkama internete, tačiau daugiausiai jos skiriasi tiktais funkcionalumu ir komandų sintakse. Duomenų vientisumo apribojimų principai yra analogiški daugumoje DBVS, kadangi duomenų saugojimo, jų unikalumo logika išlieka ta pati. Tokių sistemų palyginimui buvo pasirinktos dvi galingos DBVS – MS SQL Server ir Oracle, kurios yra vienos populiariausių stambių įmonių naudojamų duomenų bazių valdymo sistemų Lietuvoje ir užsienyje. Šių dviejų sistemų funkcionalumas yra labai panašus ir tiesa sakant, sunku būtų net pasakyti, kuri iš jų yra geresnė. Pasak literatūrinių šaltinių [8] ir [9], žiūrint į duomenų apribojimų principus, Oracle DBVS naudoja tokias pačias duomenų kontroles, kaip ir SQL Server. Tenka pripažinti, kad Oracle daugiau dėmesio skyrė trigerių apsaugai, kadangi blogai parašytas trigeris gali iššaukti nenumatytas sistemos būsenas, viena jų – amžinas ciklas. Į tokią būseną sistema gali pereiti, jeigu trigeris modifikuos lentelę, kuri ir iššaukė trigerio suveikimą. Toks klaidingai parašytas trigeris suveiks kiekvieną kartą, kai jis pats modifikuos lentelę – o tai ir sudaro ciklinį

veikimą. Oracle DBVS numatyta tokia situacija, dėl to kontroliuojamas tokių trigerių kūrimas. SQL Server ciklinė būsena pilnai įmanoma ir netgi teko su tuo susidurti.

Kitas skirtumas tarp šių dviejų DBVS yra toks, kad senesnėje SQL Server versijoje (kalbama apie SQL Server 2000) negalima sukurti trigerio, suveikiančio prieš operaciją: BEFORE INSERT, UPDATE, DELETE. Taigi pavyzdžiui norint patikrinti ar įterpinėjami duomenys yra korektiški ir tenkina sąlygas, įrašas pirmiausia bus įterpiamas į lentelę ir tik tada, jei įrašas netinkamas, lentelė gražinama į būseną, kokioje buvo prieš įterpimą, t.y. be naujo įrašo. Tokiu būdu neracionaliai eikvojami sistemos resursai, nes vietoj to, kad aptikus netinkamą įrašą neatlikti jokios operacijos, SQL Server atliks dvi operacijas – įterpimo ir pašalinimo.

Elementariausi ir patikimiausi duomenų vientisumo apribojimų realizavimo būdai yra tie, kurie yra numatyti duomenų bazių valdymo sistemose ir veikia automatiškai, be papildomų sudėtingų programinių pakeitimų. Tai yra anksčiau išanalizuotų apribojimų tipų realizacija pačioje DBVS: pirminio rakto, išorinio rakto ir reikšmės apibrėžimo (CHECK funkcija). Kiti vientisumo apribojimų realizacijos būdai yra lankstesni, tačiau sudėtingiau kuriami ir vykdomi, reikalauja daugiau darbo, laiko ir sistemos resursų.

Taigi vienas iš funkcionaliausių valdomų duomenų kontrolės mechanizmų – jau minėtas trigeris. Trigeris (angl. „*Trigger*“ – iššauti, sukelti) – tai duomenų bazės lentelei priklausantis objektas. Daugeliu savo savybių jis panašus į duomenų bazės procedūrą, skirtumas tik tas, kad pastaroji yra iškviečiama ir vykdoma programuotojo numatytu momentu, numatytoje vietoje, o trigeris „iššauna“ tik tuomet, kai lentelės įrašas yra trinamas, atnaujinamas ar įterpiamas naujas, t.y. atliekamos komandos DELETE, UPDATE ar INSERT. Kitaip tariant trigeriu galima valdyti duomenų bazės lentelės modifikavimo momentą, atlikti tam tikrus tikrinimus ir prireikus, netgi atšaukti veiksmą su lentele bei grąžinti jos buvusią duomenų struktūrą.

Duomenų bazės lentelėje modifikuojamų duomenų tikrinimas trigeryje atliekamas dviejų automatiškai sukuriamų laikinų lentelių pagalba. Kiekvienos įrašų eilutės įterpimo metu duomenų bazė suformuoja laikiną lentelę *inserted*, kurioje ir saugoma tik įterpiamoji eilutė. Jos pagalba nesunkiai galima patikrinti naujų duomenų tinkamumą ir leisti arba neleisti įrašui patekti į trigeriu kontroliuojamą lentelę. Trynimo metu, šalinamoji įrašų eilutė talpinama į kitą laikiną lentelę *deleted*, kurios struktūra yra visiškai identiška įterpimo lentelei. Įrašų eilutės atnaujinimo metu naudojamos tiek *inserted*, tiek ir *deleted* lentelės, kur pirmojoje rastume buvusio įrašo, o antrojoje naujai įterpiamo įrašo reikšmes.

Dažnai kyla klausimas, kada verta naudoti trigerius. Tai yra efektyvi ir gana išplėsta duomenų apribojimo priemonė, kuria galima kontroliuoti duomenis tiek pagal fizinę jų struktūrą (tipą, simbolių kiekį), kaip ir anksčiau analizuotuose mechanizmuose, tiek ir pagal loginę-sąlyginę

prasmę. Tačiau tai turi ir savų trūkumų, kadangi didelis vienos lentelės trigerių kiekis, gali sąlygoti duomenų bazės veikimo greitį bei stabilumą.

Tuom galima buvo įsitikinti stebint įmonės duomenų bazės veikimą ir stabilumą, kurioje yra apie 200 lentelių, kurių apimtys siekia net iki keliasdešimties milijonų įrašų, o tokios duomenų bazės duomenimis nepertraukiamai disponuoja iki 1000 vartotojų. Analizei buvo pasirinkta lentelė su 500 tūkstančių įrašų, kurios duomenis kontroliuoja 7 trigeriai. Rezultatai gana stebinantys, kadangi trigeriai įrašo įterpimą sulėtino 3-4 kartais – kas nekontroliuojamai buvo įterpiama akimirksniu, apdorojant duomenis teko laukti net iki kelių sekundžių. Taigi jeigu duomenimis reikia operuoti nepertraukiamai, prarandama nemažai laiko. Dėl to kuriant lenteles ir trigerius joms, reikia atsakingai įvertinti, iki kokio lygio reikia kontroliuoti duomenis, kad operacijos būtų atliktos optimalia sparta.

Taip pat, didelis kiekis trigerių ne tik prailgina operacijų laiką, bet ir didina tikimybę, kad jie gali trukdyti vienas kito veikimui, pavyzdžiui įterpimo metu vienas apdoros tam tikrus duomenis ir tik tada leis juos įterpti, kitas nuskaitys tuos apdorotus duomenis ir pakeis kitoje lentelėje įrašą priklausantį nuo tikrinamojo, o trečiasis trigeris nustatys, kad duomenys neatitinka tam tikros sąlygos ir nutrauks įterpimą grąžindamas viską į pirminę būseną. Natūralu, kad tokiu atveju duomenys duomenų bazėje išsikraipys ir neturės tikslumo.

Taigi galima išskirti tokius trigerių privalumus:

- patogus duomenų tikrinimas ir valdymas;
- kontrolę galima pritaikyti pagal įstaigos vidines taisykles, logikos faktorius;
- palengvina veiksmų palaikymą ir iškviečiamas reikiamu momentu;
- pagreitina programavimą, nes jie įsimenami duomenų bazėje ir nereikia kartoti kiekvienoje taikomojoje programoje, kuri susieta su duomenų baze;
- globalūs panaudojime – jeigu ta pati lentelė naudojama keliose taikomosiose programose ar jų moduliuose, visada iššaukiamas tas pats trigeris.

Deja trigerių naudojimas, turi keletą ir neigiamų aspektų, kurie ne visada leidžia pasirinkti trigerį, kaip alternatyvą:

- atliekant tikrinimus ir skaičiavimus, kažkuriam momentui sustabdomas operavimas lentele – ji užrakinama naudojimui;
- nemaža tikimybė, kad keli trigeriai prieštaraus vienas kito logikai, ko pasekoje išsikraipys duomenys;
- jeigu trigeris mėgins modifikuoti įrašą lentelėje, kuri ir sužadino jį, gali susidaryti amžinas ciklas.

Galima išvengti šių trūkumų ir sąlyginei duomenų kontrolei pasirinkti alternatyvą. Nesunku įsivaizduoti, jeigu visi skaičiavimai ir duomenų tikrinimai būtų atliekami ne duomenų bazės, o vartotojo kompiuterio lygyje, sumažėtų duomenų bazės apkrovimas, duomenų kontrolę galima būtų vykdyti ne tik tam tikriems duomenims, bet ir vartotojams arba atskiroms jų grupėms, o atliekant veiksmus, lentelės nebūtų užrakinamos naudojimui ir jomis tuo pačiu metu galėtų naudotis kiti vartotojai. Tokia alternatyva yra procedūriniai arba tiesiogiai duomenų apribojimai, kurie įterpiami kaip tikrinimo procedūros taikomosios programos moduliuose, kurie sąveikauja su tam tikra lentele. Šiuos apribojimus irgi galima valdyti pagal atliekamus veiksmus, t.y. jeigu programoje paspaustas mygtukas „Trinti“, bus atlikta duomenų kontrolė ir tik tada, jeigu sąlyga tenkinama, trynimo užklausa siunčiama į duomenų bazės valdymo sistemą. Atitinkami tikrinimai ir užklausos bus siunčiamos norint įterpti ar atnaujinti įrašą. Tai iš dalies yra patogu, nes duomenys apdorojami dar prieš kreipiantis į duomenų bazę, taigi jeigu jie netinkami, duomenų bazė net nepajudinama.

Tokiems apribojimams žymiai mažesnę įtaką turi duomenų bazės struktūra ar vartotojų skaičius, taip pat jie nėra priklausomi nuo duomenų bazės valdymo sistemos, todėl organizacijai prireikus keisti DBVS, nereikės perprogramuoti visų apribojimų, šiek tiek skirsis tik užklausų kreipinių į duomenų bazę sintaksė.

Tačiau ir šis mechanizmas nėra visapusiškai patogus naudojimui. Tarkime įmonė turi kelis šimtus kompiuterių, kuriuose įdiegta taikomoji programa dirbanti su duomenų baze. Sukūrus naują duomenų bazės lentelę, arba pasikeitus sąlygoms duomenims, naujus apribojimus tektų integruoti į kiekvieną kompiuterį atskirai. Nesunku paskaičiuoti, kad tai užimtų kur kas daugiau laiko, negu sukurti vieną naują trigerį arba pakoreguoti esamą, pritaikant jį naujai logikai. Taigi norint teisingai ir racionaliai pasirinkti tinkamiausius organizacijos duomenų vientisumo apribojimų mechanizmus, reikia labai atsakingai pasverti kas svarbiau, ar sisteminiai resursai, ar naujovių diegimas ir funkcionalumo plėtimas.

Apibendrinant išskirsiu šiuos procedūrinių arba tiesioginių duomenų apribojimų privalumus:

- neapkrauna duomenų bazės ir neužrakina lentelių naudojimui atliekant skaičiavimus;
- kontrolę galima pritaikyti ne tik tam tikriems duomenims, bet ir konkretiems vartotojams ar jų grupėms;
- didesnis apribojimų funkcionalumas, kadangi SQL kalba yra skurdesnė, nei taikomųjų programų programavimo kalbos.

Trūkumai:

- problematiškas naujų apribojimų ir sąlygų įdiegimas ar jau esamų atnaujinimas, kadangi pakeitimus reik atlikt kiekvienoje taikomojoje programoje;

- nepakankamas suderinimas su DBVS;
- padidėja tinklo apkrovimas, kadangi kontroliuojant duomenis, kiekviena taikomoji programa papildomai kreipiasi į duomenų bazę.

Paskutinysis analizuotas duomenų apribojimo realizavimo būdas – veiklos taisyklių saugyklos. Tai iš ties funkcionalus ir naudingas įrankis, kadangi veiklos taisyklės sudaromos dar organizacijos IS projektavimo metu, taigi jas galima pritaikyti ir tolimesniuose etapuose. Veiklos taisyklėmis galima itin konkrečiai apibrėžti duomenis, kuriais operuojama. Tačiau jų negalime traktuoti, kaip duomenų bazės trigeriai ar kiti apribojimai, tai visiškai atskiras modeliavimo objektas, naudojamas kur kas platesnėje srityje nei tik duomenų vientisumo užtikrinimui. Tai dar palyginus nauja metodologija, dėl to yra labai nedaug duomenų bazių valdymo sistemų, kur ji būtų realizuota.

Daugumos apribojimų mechanizmų veikimų analizė buvo atliekama naudojantis MS SQL Server 2000 duomenų bazės valdymo sistema. Deja su Oracle dirbti neteko, dėl to apie jos funkcionalumą ir duomenų apribojimų mechanizmus plačiau analizuota tik iš literatūrinių šaltinių [9], [11]. Apibendrinimui visus duomenų vientisumo apribojimų tipų realizacijos būdus galima suvesti į lentelę.

2 lentelė. Apribojimų realizacijos būdai

Apribojimo realizacijos būdas	**Privalumai**	**Trūkumai**
Lentelės laukų taisyklės: • būtina reikšmė; • reikšmės apribojimas	• griežtai apriboja reikšmes; • yra taikomos visoms lentelėms, taigi lentelė visada turės apribojimą;	• mažas apribojimų lankstumas; • sunku arba išvis neįmanoma apibrėžti įmonės taisykles, kritines duomenų reikšmes, sulyginimą su kita lentelės reikšme.
Pirminis raktas	• neleidžia dubliuotis įrašams; • ženkliai paspartina raktinio lauko reikšmės paiešką.	
Išorinis raktas	• paspartina raktinių laukų reikšmių paiešką tarp lentelių.	• jei išorinis raktas neturi kitos lentelės pirminio rakto atitikmens, pažeidžiamas DB nuorodų vientisumas
Trigeris	• patogus duomenų tikrinimas ir valdymas; • kontrolę galima pritaikyti pagal savus logikos faktorius; • palengvina veiksmų palaikymą ir iškviečiamas reikiamu	• atliekant tikrinimus ir skaičiavimus, laikinai sustabdomas operavimas lentele – ji užrakinama naudojimui; • nemaža tikimybė, kad keli trigeriai prieštaraus vienas kito

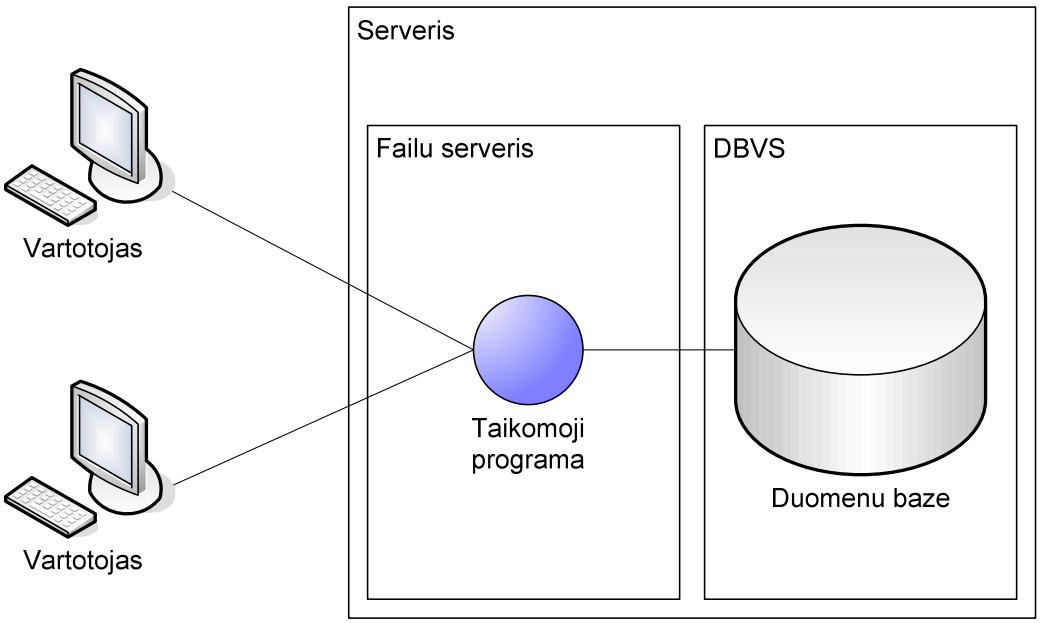
	momentu; • pagreitina programavimą, nes jie įsimenami pačioje duomenų bazėje; • jeigu ta pati lentelė modifikuojama keliose taikomosiose programose ar jų moduliuose, visada iššaukiamas tas pats trigeris.	logikai, ko pasekoje bus iškraipyti duomenys; • jeigu trigeris modifikuos įrašą lentelėje, kuri jį sužadino, gali susidaryti amžinas ciklas.
Apribojimų valdiklis taikomojoje programoje	• neapkrauna duomenų bazės ir neužrakina lentelių naudojimui atliekant skaičiavimus; • kontrolę galima pritaikyti ne tik tam tikriems duomenims, bet ir konkretiems vartotojams ar jų grupėms.	• problematiškas naujų apribojimų ir sąlygų įdiegimas ar jau esamų atnaujinimas; • nepakankamas suderinimas su DBVS; • padidėja tinklo apkrovimas, kadangi kontroliuojant duomenis, kiekviena taikomoji programa papildomai kreipiasi į duomenų bazę.

2.3. Eksperimentinio tyrimo uždaviniai

Analizė rodo, kad organizacijos gali pasirinkti įvairius vientisumo apribojimų realizavimo variantus:

- Visus apribojimus realizuoti duomenų bazėje;
- Visus apribojimus realizuoti taikomojoje programoje;
- Realizuoti apribojimus ten, kur jų veikimas efektyvesnis – dalį duomenų bazėje, dalį – taikomojoje programoje.

Pastarajam atvejui ištirti ir skirtas tolesnis tyrimas. Šiam tyrimui skirtos sistemos architektūra pavaizduota 1 paveikslėlyje. Atsižvelgiant į duomenų vientisumo apribojimų realizacijos būdų trūkumus ir privalumus, pagal ją bus suprojektuota prototipinė įmonės informacinė sistema.



1pav. Tiriamos sistemos architektūros schema

Galimas ir toks atvejis, kai organizacija naudoja keletą paskirstytų duomenų bazių, kurioms taikomi tie patys apribojimai, ir turi apribojimų realizacijas tiek taikomosios programos, tiek duomenų bazės priemonių pavidale ir pasirenka vieną ar kitą būdą konkrečios duomenų bazės metu.

Taigi šio tyrimo uždaviniai yra:

- Išanalizuoti duomenų bazių valdymo sistemos duomenų vientisumo apribojimų tipus ir jų realizavimo principus.
- Ištirti apribojimų realizavimo DB ir taikomosiomis programomis privalumus bei trūkumus įvairių apribojimų tipų atvejais.
- Sudaryti vientisumo apribojimų realizavimo rekomendacijas ir metodiką.
- Pritaikyti ją konkrečiai informacinei sistemai.

2.4. Vientisumo apribojimų realizavimo kriterijai

Stambioms organizacijoms su didelėmis duomenų bazių sistemomis labai svarbi yra duomenų darna ir vientisumas. Tačiau kad tai užtikrinti, reikia įvertinti ne tik duomenų kontrolės priemones, bet ir sistemos resursus bei klaidų tikimybes. Dėl to siekiama surasti optimalų duomenų vientisumo apribojimų mechanizmą, kuris eikvotų minimalų kiekį sistemos resursų ir užtikrintų maksimalią duomenų vientisumo kontrolę.

Sistemos funkcijos

- Sistema turi užtikrinti duomenų vientisumą ir darną;

- Sistema turi informatyviai pranešti vartotojui apie galimus ar jau įvykusius duomenų vientisumo pažeidimus.

Kadangi tai yra reliacinė duomenų bazė, duomenys, su kuriais operuoja sistema, yra tekstinio formato, tačiau jų tipas, dydis ir sudėtingumas gali būti įvairus, dėl to ir reikalinga jų kontrolė vientisumui užtikrinti.

Nefunkciniai reikalavimai ir apribojimai

- Sistema, kurioje veikia duomenų apribojimo priemonės turi būti stabili;
- Operavimas duomenimis turi būti spartus;
- Sistema privalo veikti visiems duomenų bazės vartotojams.

Yra kelios galimos rizikingos situacijos sistemos veikimo metu. Viena jų tokia, kad jeigu bus iššauktas amžinas ciklas, sutriks sistemos stabilumas. Kita – neteisingai suformavus apribojimus, gali būti sugadinti arba prarasti svarbūs duomenys. Pats rezultatas laikomas kokybiškas tik tuomet, kai duomenų apribojimai veikia efektyviausiai su mažiausiomis sistemos resursų sąnaudomis.

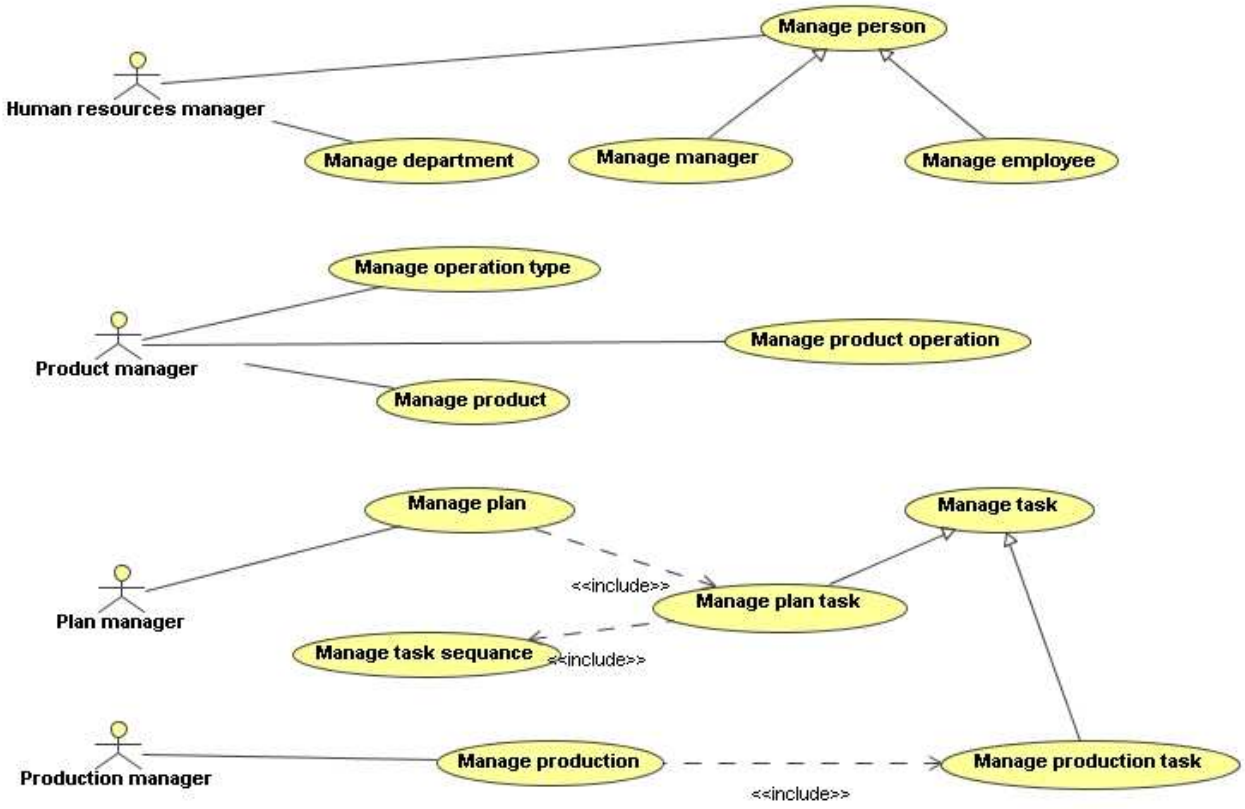
2.5. Analizės išvados

- Ištyrus trigerių veikimą, paaiškėjo, kad tai funkcionali, tačiau daug sistemos resursų eikvojanti priemonė. Taip pat naudojantis jais yra nemaža tikimybė iššaukti amžiną ciklą arba duomenų iškraipymą.
- Praktika rodo, kad nevertėtų lentelei kurti daugiau nei 2-3 trigerių. Vietoj to geriau naudoti procedūrinius apribojimus, veikiančius taikomojoje programoje, per kurią sąveikaujama su duomenų baze.
- Norint rasti tinkamiausią didelės sistemos duomenų vientisumo apribojimų realizacijos strategijos parinkimo metodiką, reikia atlikti tyrimą eksperimentinėje sistemoje, realizuojant duomenų vientisumo apribojimų tipus tiek trigeriais, tiek apribojimų valdikliais, tiek ir kombinuojant šiuos abu apribojimų realizavimo būdus tarpusavyje, ir sulyginti gautus jų veikimo charakteristikų matavimų rezultatus.

3. Eksperimentinės informacinės sistemos su vientisumo apribojimais pavyzdys

Apribojimams realizuoti ir tirti reikia pasirinkti sistemą, kuri būtų aiškiai apibrėžta ir žinomi jos reikalavimai. Šiam darbui pasirinkta gana sudėtinga sistema, kurios tikslas – registruoti gamybinės sistemos produktus, jų gaminimo operacijas, gamybos planus, užduotis, paskirti vykdytojus ir t.t. . Reikalavimai šios sistemos funkcionalumui pateikiami tolesniuose poskyriuose.

3.1. Gamybinės sistemos reikalavimų specifikacija



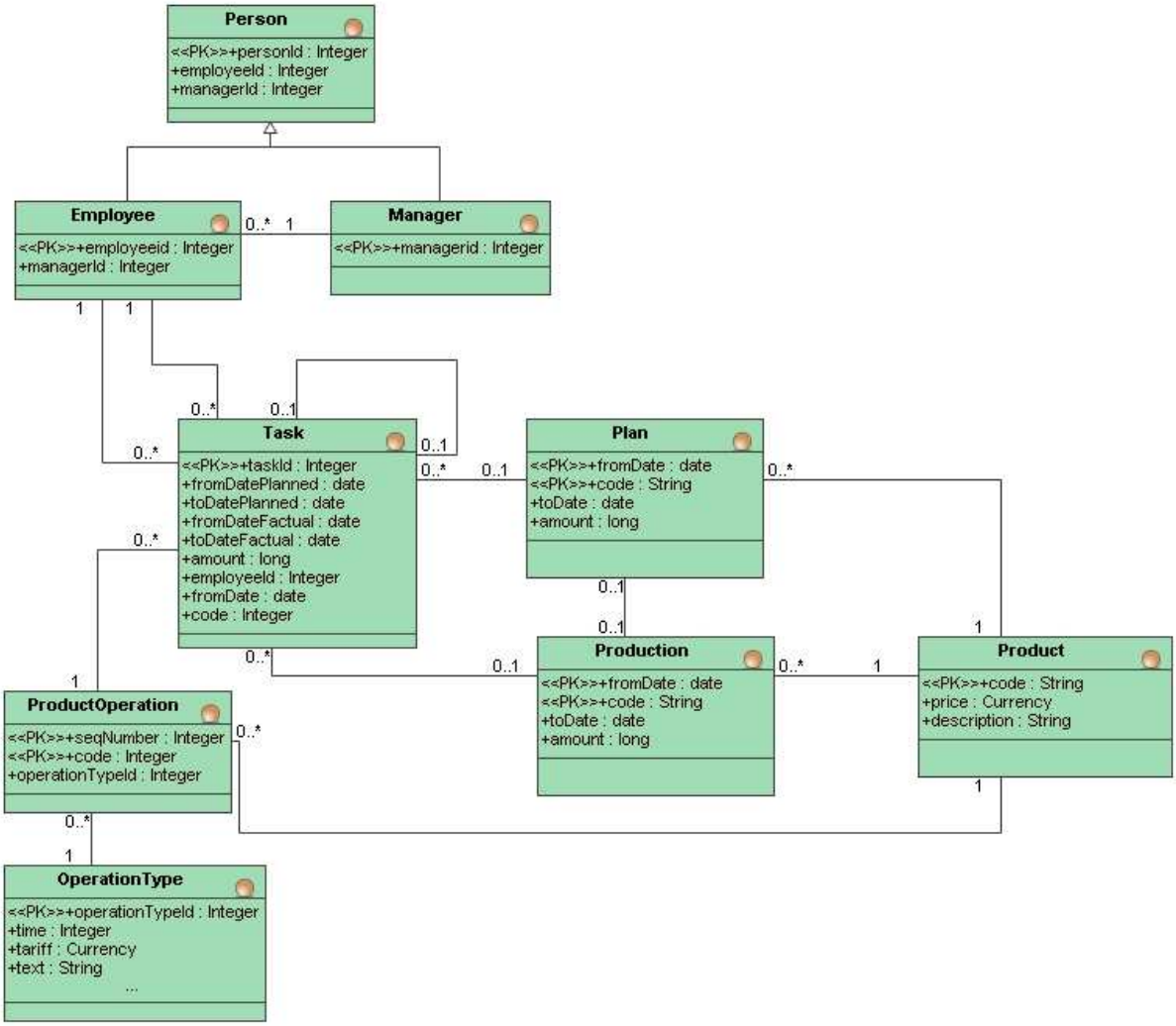
2 pav. Sistemos vartotojų panaudojimo atvejų diagrama

Taigi apribojimų realizavimui pasirinkta gamybos sistema. Joje gaminami produktai (Product), kurių kiekvienam pagaminti reikalinga operacijų seka (ProductOperation). Operacijų sekas sudaro nustatytų tipų operacijos (OperationType). Produktus galima gaminti pagal planą (Plan) arba be plano, vykdant gamybos užduotis (Task). Jei gaminama pagal planą, gamybos vykdymas (Production) ir užduotys turi ryšį su planu. Su gamyba susiję asmenys skirstomi į vadybininkus (Manager) ir darbuotojus (Employee). Užduotis susijusi su dviem darbuotojais: vykdytoju (Performer) ir tikrintoju (Inspector). Tai gana bendros paskirties modelis, kuris gali

vaizduoti tiek tikrą produktų gamybą, tiek įvairiausių kitų rūšių veiklą, pavyzdžiui, projektavimą, programavimą, leidybą ir pan.

3.2. Dalykinės srities modelis

Šios sistemos dalykinės srities modelis (esybių klasių diagrama) atrodo taip:



3 pav. Esybių klasės diagrama

Joje matome pagrindinius sistemos objektų tipus, kurie pagal nurodytus ryšius ir kardinalumą vienaip ar kitaip įtakoja viens kitą. Generuojant duomenų bazės schemą, apibendrinimo ryšį reikia pakeisti paprasta asociacija su kardinalumu 1 → 0..1.

Kaip matome sistema yra pakankamai nemaža ir sudėtinga, o sistemos reikalavimai apima ne tik taikomosios programos veikimo lygį, bet ir duomenų vientisumo apribojimus tiek duomenų bazėje, tiek ir vartotojo kompiuteryje. Taigi kuriant tokią sistemą, reikalingas tinkamas šių apribojimo strategijų suderinimas.

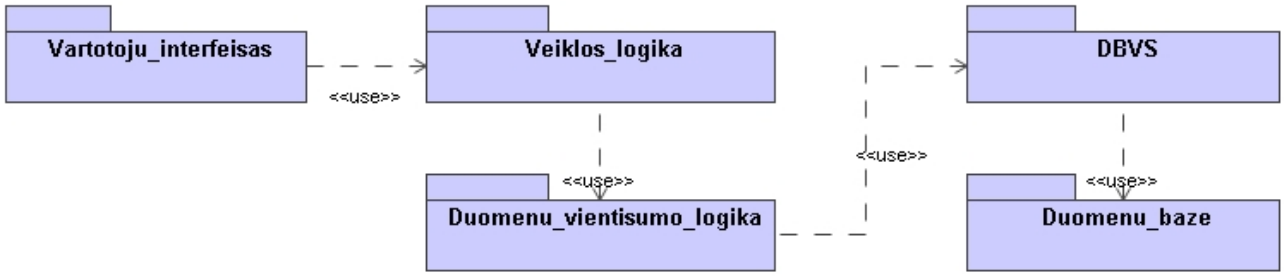
3.3. Eksperimentinės sistemos realizacijos pasirinkimas

Apribojimų įgyvendinimo modelis pasirinktas, atsižvelgiant į tai, kad šie apribojimai funkcionuos gana didelėje duomenų bazėje, su šimtais lentelių ir milijonais įrašų jose. Operuojant tokiomis duomenų bazėmis, labai svarbu pasirinkti tinkamą duomenų mainų ir apribojimų strategiją. Iš praktikos pastebėta, kad duomenų apribojimus realizuojant tik duomenų bazės lygyje, sulėtėja darbas jose, prarandama nemažai laiko resursų vykdant užklausas, o kartais sutrinka ir pačios duomenų bazės veikimas. Tokias problemas įmanoma sumažinti, neprarandant duomenų kokybės, tam tikrus duomenų apribojimus padalinant duomenų bazės ir su ja tiesiogiai komunikuojančios taikomosios programos lygyje. Šis spendimas ypatingai aktualus, jei reikalinga atlikinėti sudėtingus finansinius, gamybinius, mokslinius skaičiavimus ir pan. Daugumą skaičiavimo operacijų perkėlus į vartotojo kompiuterį, ar serverį, kuris yra atskirtas nuo duomenų bazės serverio, sumažinamas veikimo laikas.

Mažose sistemose, toks duomenų apribojimo modelis neduos akivaizdžių rezultatų, dėl to su pasirinkta eksperimentine duomenų baze sunku realiai atskleisti metodikos pranašumą, o išvados gali būti daromos labiau teorinės. Tačiau šis tyrimas paremtas praktinėmis žiniomis ir jų pritaikymas pasirinktai dalykinei sričiai leis susidaryti apribojimų realizacijos strategijos bendrą vaizdą.

3.4. Vientisumo apribojimų realizacijos architektūra

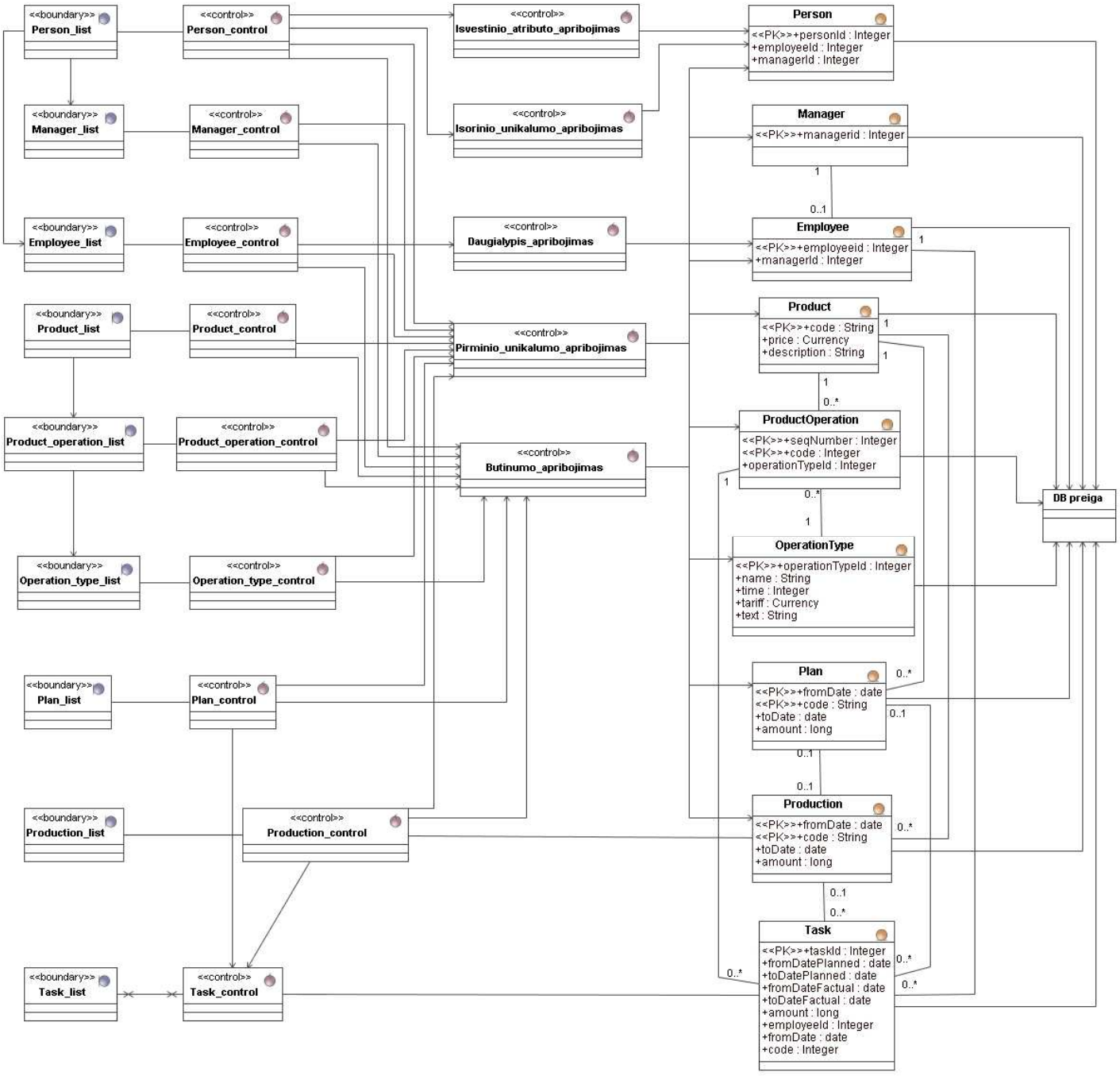
Informacinės sistemos su vientisumo apribojimais loginė architektūra pavaizduota paveikslėlyje:



4 pav. Trijų lygių sistemos architektūra

Jos esmė yra tokia, kad veiklos srityje yra papildomas duomenų valdiklis, tam tikros apdorojimo ir tikrinimo funkcijos, kurios yra alternatyvos kai kuriems duomenų bazės duomenų apribojimams.

Smulkiau detalizuojant kiekvieno lygio architektūrą, gaunama tokia sistemos schema:

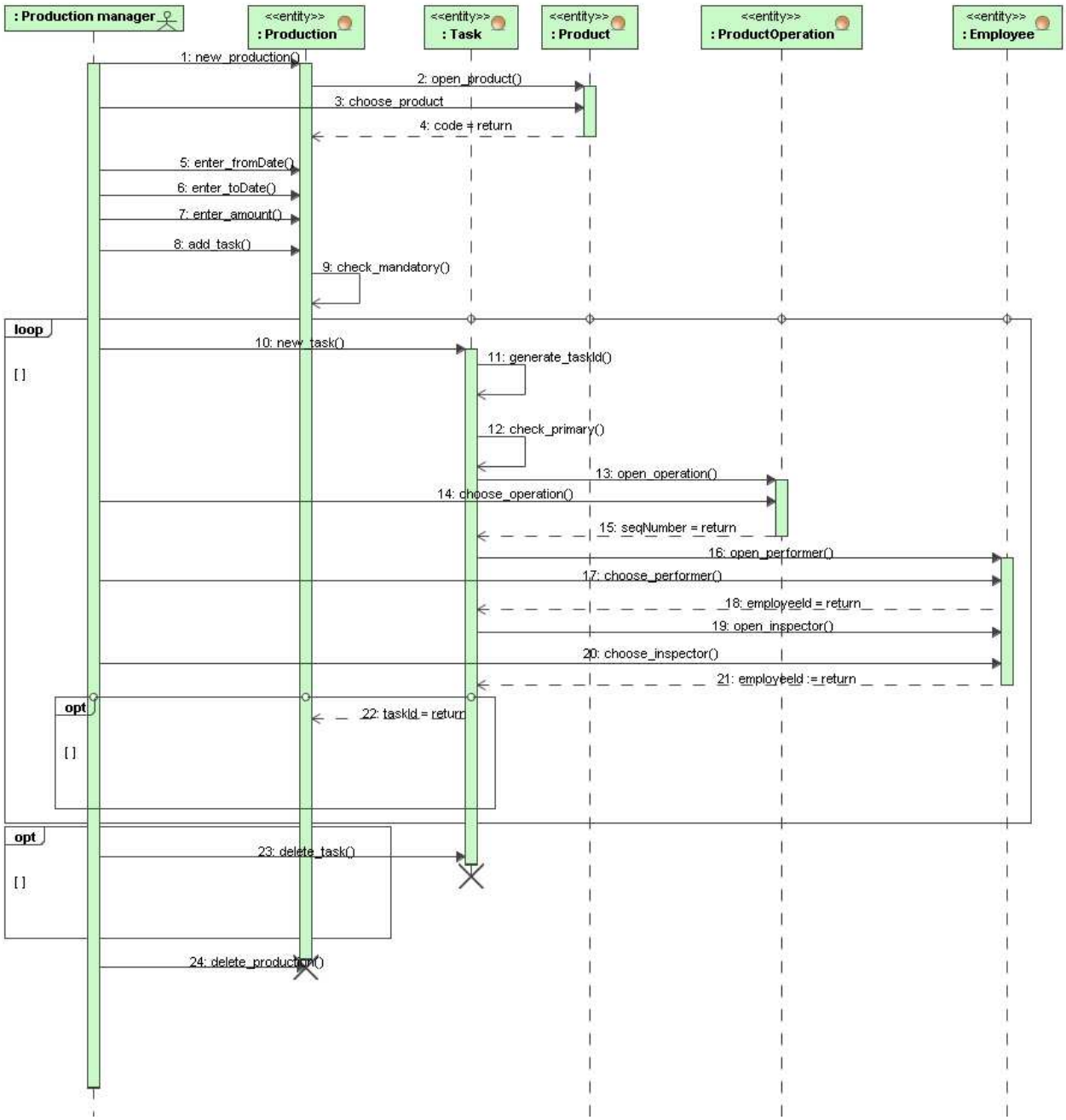


5 pav. Sistemos trijų lygių klasių diagrama

3.5. Sistemos elgsenos modelis

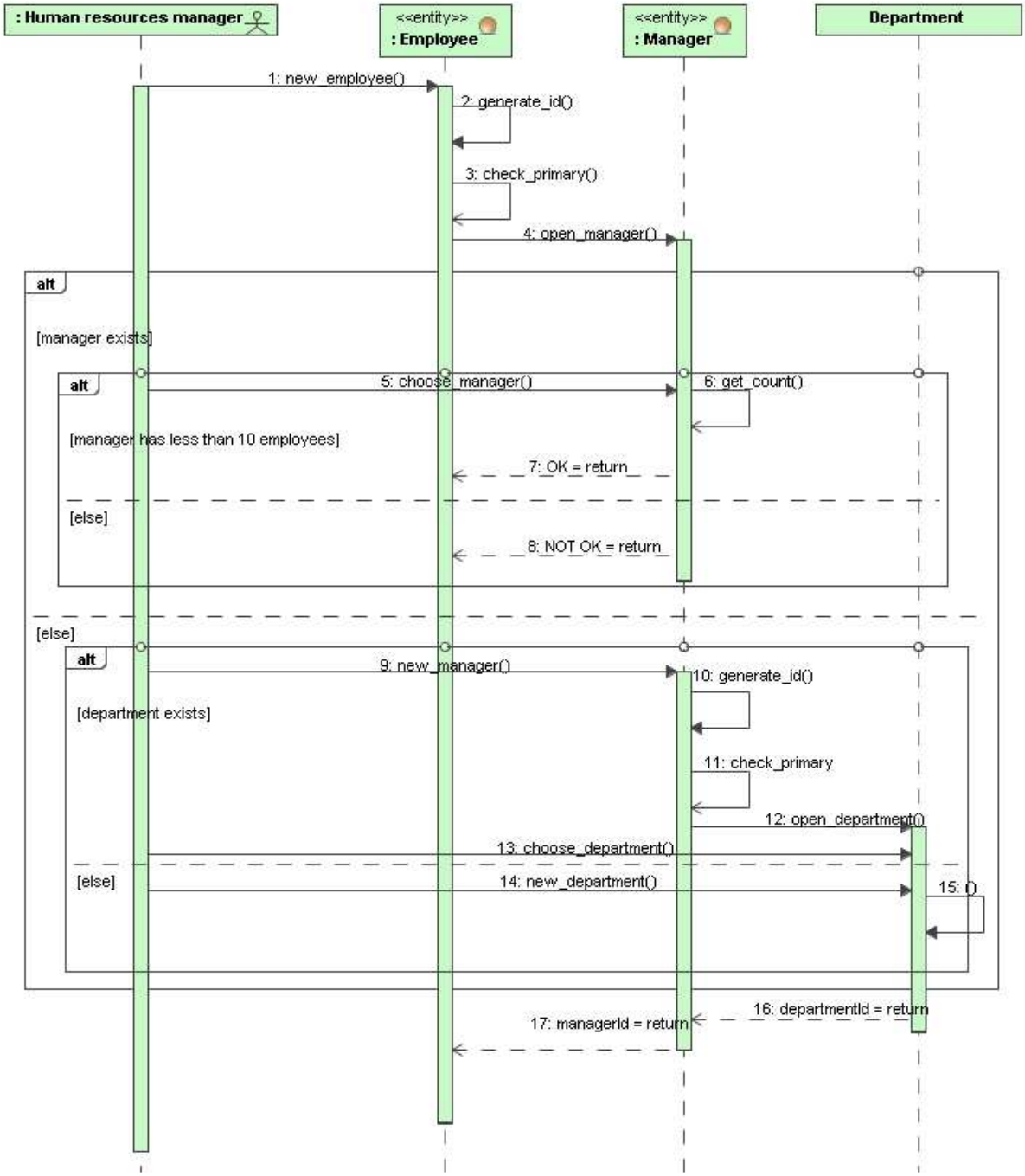
Sistemos elgsenos modelį gerai atspindi sekų diagrama. Joje matomos vykdomos operacijos ir dalyvaujantys sistemos elementai. Pavaizduosime dviejų pagrindinių sistemos panaudojimo atvejų sekų diagramas, kurios atspindi svarbiausius sistemos veikimo principus.

Pirmoji sekų diagrama vaizduoja produkto gamybos procesą, valdomą užduotimis:



6 pav. Produkto gamybos pagal užduotis sekų diagrama

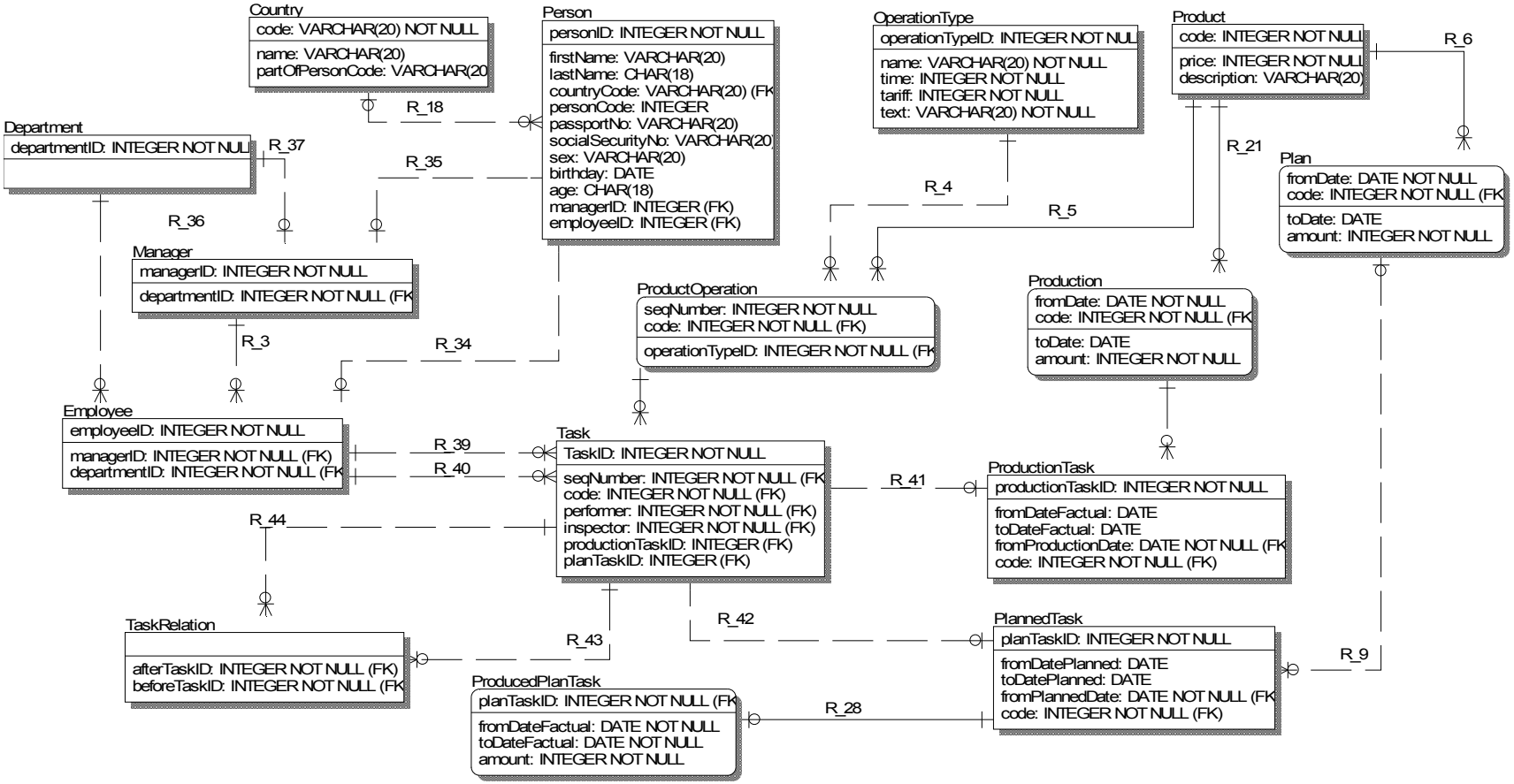
Antroji sekų diagrama parodo sistemos dalyvių kūrimo procesą:



7 pav. Personalo valdymo sekų diagrama

3.6. Duomenų bazės schema

Tiriant apribojimų realizavimą, iš dalykinės srities modelio sudaryta duomenų bazės schema, kuri pateikta paveikslėlyje:



8 pav. Duomenų bazės modelis

3.7. Vientisumo apribojimų realizacijos tipai

Šiame skyriuje pateikti įvairių tipų duomenų apribojimų realizavimo modeliai, pasirinktai dalykinei sričiai. Šių apribojimų yra gana nemažai, kai kurie jų yra pasirenkami ar automatizuoti pačiose duomenų bazėse, kiti programuojami pagal tam tikrus poreikius ar aplinkybes. Visi duomenų bazės lygio apribojimai realizuoti MS SQL sintakse, o pati taikomoji programa komunikuojanti su duomenų baze – MS FoxPro programavimo kalbos sintakse.

3.7.1. Pirminio rakto apribojimas

Pirminis raktas yra naudojamas lentelės lauko raktui nusakyti. Jis neleidžia atsirasti dviems vienodoms lauko reikšmėms ir reiškia lauką, pagal kurį pilnai galima identifikuoti įrašą. Mūsų duomenų bazės modelyje, kiekviena lentelė turi bet vieną arba du šiuos apribojimus. Jie buvo realizuoti lentelių kūrimo metu. Bendra SQL sintaksė šiam apribojimui:

```
ALTER TABLE <lenteles_pavadinimas>
ADD CONSTRAINT <apribojimo_pavadinimas>
PRIMARY KEY (<lauko_pavadinimas1> [, < lauko_pavadinimas2>...])
```

Lentelei PLAN pirminio rakto apribojimo sukūrimo SQL sintaksė yra tokia:

```
ALTER TABLE [dbo].[Plan]
ADD CONSTRAINT [PK_Plan]
PRIMARY KEY ([fromDate], [code])
```

Lentelei PERSON:

```
ALTER TABLE [dbo].[Person]
ADD CONSTRAINT [PK_Person]
PRIMARY KEY ([personID])
```

Likusiose lentelėse šis apribojimas realizuotas analogiškai. Reikia atkreipti dėmesį į lentelę TASKRELATION, kadangi schemoje šiai lentelei pirminis raktas neparodytas, bet tai nereiškia, kad ši lentelė jo neturi. Pirminis identifikatorius joje sukurtas lauke, kuris įgyja automatiškai didėjančią reikšmę (autoincrement), t.y. to lauko reikšmė nėra aktuali kitiems duomenų bazės objektams, niekam neįdomu kokia ji, tačiau ji reikalinga, kadangi kiekvienoje lentelėje reikalingas identifikacinis laukas, pagal kurį vienas įrašas skirtųsi nuo kitų lentelės įrašų.

3.7.2. Atributo vidinio unikalumo identifikatorius

Atributo unikalumo identifikatorius yra naudojamas lentelės lauko unikalumui nusakyti. Jis kaip ir pirminio rakto apribojimas neleidžia lentelėje atsirasti besidubliuojančioms to lauko reikšmėms. Šį apribojimą galima sukurti lentelės kūrimo arba apribojimų papildymo metu sukūrus apribojimą UNIQUE. Apibendrintas apribojimo sukūrimas lentelės papildymo SQL sintakse atrodo taip:

```
ALTER TABLE <lenteles_pavadinimas>
ADD CONSTRAINT <apribojimo_pavadinimas>
UNIQUE  (<lauko_pavadinimas1> [, < lauko_pavadinimas2>...])
```

Lentelės COUNTRY lauko *name* reikšmės turi būti unikalios, taigi lentelę reikėtų papildyti nauju apribojimu:

```
ALTER TABLE [dbo].[Country]
ADD CONSTRAINT [IX_Country]
UNIQUE ([name])
```

3.7.3. Išorinis atributo unikalumo apribojimas

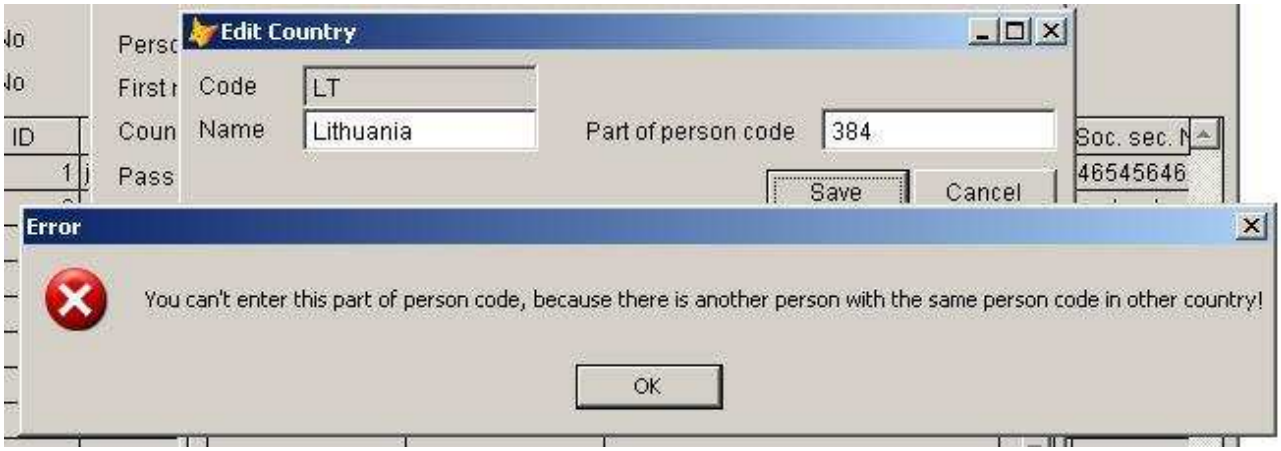
Išorinio unikalumo apribojimas naudojamas, kai reikia identifikuoti keliose lentelėse susietus egzistuojančius atributus. Šio apribojimo realizacijai atlikti naudojami trigeriai lentelėse, kuriose šis apribojimas yra taikomas. Jų kūrimo šablonas atrodo taip:

```
CREATE TRIGGER <trigerio_pavadinimas> ON <lentelės_pavadinimas>
FOR OF INSERT, UPDATE
AS
BEGIN
        <programinis_SQL_kodas>
END
```

Tiriamoje duomenų bazėje išorinio atributo apribojimo reikia lentelėms COUNTRY ir PERSON, kadangi pastarojoje lentelėje gali pasitaikyti atvejis, kai asmenys yra iš skirtingų šalių ir turi tą patį asmens kodą. Tačiau šiuo atveju trigeriai nebus naudojami, o išorinis unikalumas tikrinamas dar taikomojoje programoje sukurta funkcija su parametrais *check_unique(old_partOfPersonCode,new_partOfPersonCode).* Toks apribojimo realizacijos būdas pasirinktas todėl, kad užklausos pagalba patikrinus reikiamus įrašus, jeigu sąlyga netenkinama, išvengiama papildomos užklausos į duomenų bazę. Šiuo atveju bus nevykdoma įterpimo užklausa.

Šį apribojimą realizuojant trigeriu, pirmiausia programa siųstų įterpimo užklausą duomenų bazei, kuri iššauktų trigerio suveikimą. Trigeryje būtų tikrinami tie patys duomenys kaip ir programos funkcijoje, ir, jei sąlyga netenkinama, atšaukiama įterpimo užklausa. Taigi matome, kad mūsų pasirinktu realizacijos būdu laimime siųsdami viena užklausa mažiau. Funkcijos programinis kodas:

Šis apribojimas reikalingas tik atnaujinant jau įvestus duomenis. Funkcija pagal įvestą asmens kodo dalį (COUNTRY laukas *partOfPersonCode*) užklausos pagalba atsirenka šalis ir tikrina ar asmenys, kurie yra iš šių šalių, neturi vienodų asmens kodų. Radus šio apribojimo pažeidimą, parodomas pranešimas, o operacijos atšaukiamos:



9 pav. Išorinis atributo unikalumo apribojimas

Funkcijos kodas MS FoxPro sintakse:

```
FUNCTION check_unique
LPARAMETERS old, new
LOCAL SQL_
SQL_ = "SELECT PERSONCODE FROM COUNTRY C1 "+;
        "INNER JOIN PERSON ON COUNTRYCODE = C1.CODE "+;
        "WHERE C1.PARTOFPERSONCODE = ?NEW AND PERSONCODE IN ( "+;
        "       SELECT PERSONCODE FROM COUNTRY C2 "+;
        "       INNER JOIN PERSON ON COUNTRYCODE = C2.CODE "+;
        "       WHERE C2.PARTOFPERSONCODE = ?OLD )"
SQLEXEC(hConn,SQL_,"result")
IF RECCOUNT("result") > 0
  MESSAGEBOX("You can't enter this part of person code, because there is another person with the same person code in other country",16,"Error")
  RETURN .F.
ENDIF
```

```
RETURN .T.
```

Nors šį apribojimą realizuojant vistiek kreipiamasi į duomenų bazę, tačiau išvengiame vaizdų ir gremėzdiškų trigerių kūrimo.

Tokiu pačiu principu galima realizuoti ir PERSON lentelės išorinio unikalumo apribojimą. Šiuo atveju įterpiant naują arba atnaujinat seną įrašą, reikalinga patikrinti ar nėra kitų šalių, su vienodais PartOfPersonCode, kur jau egzistuoja toks asmens kodas. Apribojimui realizuoti, naudosime panašią funkciją *check_personcode(personCode, countryCode)* ir, jeigu ji grąžins reikšmę didesnę už 0, vadinasi asmens kodai PERSON lentelėje dubliuojasi. Funkcijos programinis kodas:

```
FUNCTION check_personcode
LPARAMETERS pcode, ccode
LOCAL sql_
SQL_ = "SELECT P1.PERSONCODE FROM PERSON P1 "+;
  "INNER JOIN COUNTRY C1 ON P1.COUNTRYCODE = C1.CODE "+;
  "WHERE P1.PERSONCODE = ?pcode AND P1.COUNTRYCODE = ?ccode
  “C1.PARTOFPERSONCODE IN ( "+;
  "         SELECT P2.PARTOFPERSONCODE FROM PERSON P2 "+;
  "         INNER JOIN COUNTRY C2 ON P2.COUNTRYCODE = C2.CODE "+;
  "         WHERE P2.PERSONCODE = ?pcode AND "+;
  "          P2.COUNTRYCODE != ?ccode)"
SQLEXEC(hConn,sql_,”count”)
IF RECCOUNT(“count”) > 0
  MESSAGEBOX("You can't enter this person code, because there is another person with the same person code in other country",16,"Error")
  RETURN .F.
ENDIF
RETURN .T.
```

3.7.4. Išorinio rakto apribojimas

Šis apribojimas yra naudojamas norint susieti lenteles tarpusavio ryšiu. Jis kuriamas lentelės laukui, kuris turi būti susietas su kitos lentelės pirminiu lauku. Bendra SQL sintaksė šio apribojimo sukūrimui:

```
ALTER TABLE <lenteles_pavadinimas1>
```

ADD CONSTRAINT <apribojimo_pavadinimas>
FOREIGN KEY (<lauko_pavadinimas1> [, < lauko_pavadinimas2>...])
REFERENCING <lenteles_pavadinimas2> (<lauko_pavadinimas3> [, <lauko_pavadinimas4>...])

Toks apribojimas labai praverčia, kai norima laikytis tam tikros įvykių sekos, pavyzdžiui į lentelę negali būtį įvesta reikšmė, kol ji nesukurta kitoje lentelėje, arba apsaugant duomenų ryšius pašalinant įrašą, tuomet pašalinimas nebus vykdomas, jeigu ši reikšmė egzistuoja kitoje lentelėje, kuri siejasi per nurodomąjį atributo apribojimą.

Pasirinktos duomenų bazės modelyje šis apribojimas reikalingas daugeliui lentelių. Viena jų yra EMPLOYEE lentelė, kurios laukai *managerID* ir *departmentID* turi išorinio rakto apribojimą (t.y. turi ryši atitinkamai su lentelės MANAGER lauku *managerID* ir lentelės DEPARMENT lauku *departmentID*). Tokiu atveju negalima lentelėje EMPLOYEE į šiuos laukus įvesti reikšmių, kol tokie įrašai nebus sukurti MANAGER ir DEPARTMENT lentelėse. Žinoma ir atvirkščiai – negalima pašalinti įrašų iš pastarųjų lentelių, kol egzistuoja įrašai su tokiomis reikšmėmis EMPLOYEE lentelėje. Išorinio rakto apribojimas šiai lentelei kuriamas tokia SQL sintakse:

```
ALTER TABLE [dbo].[Employee]
ADD CONSTRAINT [FK_Employee_Department]
FOREIGN KEY ([departmentID])
REFERENCES [dbo].[Department] ([departmentID]),
CONSTRAINT [FK_Employee_Manager]
FOREIGN KEY ([managerID])
REFERENCES [dbo].[Manager] ([managerID])
```

Lentelės PLAN laukui *code* toks apribojimas taipogi reikalingas, kadangi negalima sukurti plano produkto gamybai, kol nežinoma, koks gaminamas produktas:

```
ALTER TABLE [dbo].[Plan]
ADD CONSTRAINT [FK_Plan_Product]
FOREIGN KEY ([code])
REFERENCES [dbo].[Product] ([code])
```

Kitų lentelių laukams šis apribojimas sukuriamas analogiškai. Galima pastebėti tai, kad lentelės TASK visi ne pirminio identifikavimo laukai turi išorinio rakto apribojimą, t.y. iš bet kokios jos reikšmės, galime susirasti konkretų įrašą kitoje lentelėje, su kuria ji siejasi. Lentelės

TASKRELATION abu ne pirminio identifikavimo laukai taip pat turi išorinio rakto apribojimą, tačiau jie siejasi su tos pačios TASK lentelės lauku *taskId*, norint sukurti refleksyvinį ryšį.

3.7.5. Būtinasis atributo apribojimas

Šis apribojimas nurodomas lentelės kūrimo procese laukui, kuris privalo turėti bent kokią reikšmę (ne tuščią). Tokių laukų reikšmės dažniausiai yra svarbios kituose sistemos procesuose. Pirminio rakto laukai būtinąjį apribojimą įgyja automatiškai, sukuriant patį raktą, kadangi tuščia reikšmė negali identifikuoti įrašo. Apibendrinta SQL sintaksė šiam apribojimui yra lentelės kūrimo komanda, su nurodytu papildomu parametru:

```
CREATE TABLE <lenteles_pavadinimas1>
(
<lauko_pavadinimas1> <duomenu_tipas> NOT NULL,
<lauko_pavadinimas2> <duomenu_tipas> NULL
)
```

Jeigu yra nurodytas parametras NULL, vadinasi šio lauko reikšmės gali turėti bet kokią, net ir tuščią reikšmę. Nenurodžius jokio parametro, pagal nutylėjimą lentelei sukuriamas laukas su NULL parametru.

Lentelės OPERATIONTYPE laukams būtinumo apribojimas kuriamas taip:

```
CREATE TABLE [dbo].[OperationType]
(
[operationTypeID] [int] NOT NULL,
[name] [varchar] (20) NOT NULL,
[time] [int] NOT NULL,
[tariff] [int] NOT NULL,
[text] [varchar] (20) NOT NULL
)
```

Šios lentelės visų laukų reikšmės yra būtinos (ne tuščios):

- operacija būtinai turi turėti savo pavadinimą, kitaip tik pagal jos kodą sunku bus atpažinti, kokia tai operacija;
- būtinai reikia žinoti ir operacijos trukmę;
- turint trukmę ir apmokėjimo tarifą, bus paskaičiuoti operacijos kaštai.

Reikia atkreipti dėmesį kad pirminio rakto laukui *operationTypeID* taip pat nurodytas būtinumo apribojimas, kadangi pirminio rakto apribojimas sukuriamas vėliau nei pati lentelė.

Lentelei PRODUCT nesvarbus produkto aprašymas, tačiau yra būtinai reikalingas kodas (pirminis raktas) ir produkto kaina:

```
CREATE TABLE [dbo].[Product]
(
[code] [int] NOT NULL,
[price] [int] NOT NULL,
[description] [varchar] (20) NULL
)
```

Šį apribojimą galima realizuoti ir pačioje taikomojoje programoje, tikrinant ar į laukelius buvo įvestos nors kokios reikšmės. Taip išvengiama bereikalingų kreipinių į duomenų bazę, jeigu ir taip žinoma, kad įrašas nebus įterpiamas į lentelę, nes yra nenurodyta (NULL) reikšmė. Gamybos valdymo taikomojoje programoje suveikus šiam apribojimui parodomas pranešimas, o operacijos atšaukiamos:

10 pav. Būtinasis atributo apribojimas

3.7.6. Atributo reikšmės apribojimas

Šis atributo reikšmės apribojimas naudojamas siekiant patikrinti vienokio ar kitokio lauko reikšmę. Tai gali būti:

- konkrečiai įvardintos lauko reikšmės;
- apibrėžtas reikšmių intervalas;
- netuščios reikšmės apibrėžimas (NOT NULL)
- mišrus apribojimas naudojant šių apribojimų derinius.

Bendra MS SQL sintaksė šiam apribojimui:

```
ALTER TABLE <lenteles_pavadinimas1>
```

```
ADD CONSTRAINT <apribojimo_pavadinimas>
CHECK (<lauko_pavadinimas1> BETWEEN <reikšmė1> AND <reikšmė2>
AND  <lauko_pavadinimas1>  <> <reikšmė3>)
```

Arba jei apribojamos nebūtinos laukų reikšmės:

```
ALTER TABLE <lenteles_pavadinimas1>
ADD CONSTRAINT <apribojimo_pavadinimas>
CHECK (<lauko_pavadinimas1> IS NOT NULL AND <lauko_pavadinimas2> IS NULL
        OR  <lauko_pavadinimas1> IS NULL AND <lauko_pavadinimas2> IS NOT NULL)
```

Šis apribojimas pritaikytas daugeliui pasirinktos dalykinės srities duomenų bazės objektų. Vienas jų yra lentelė PERSON ir jos lauko *sex* apribojimas reikšmėms „M“ (Male) ir „F“ (Female), t.y. šio lauko įrašo reikšmė galės turėti tiktai tokias apibrėžtas reikšmes. Apribojimo lentelei SQL sintaksė:

```
ALTER TABLE [dbo].[Person]
ADD CONSTRAINT [CK_Person_1]
CHECK ([sex] = 'F' or [sex] = 'M')
```

Taip pat šis apribojimas reikalingas ir kitiems lentelės PERSON laukams *employeeID* ir *managerID*, kadangi kiekvienas įrašas turi turėti nenulinę reikšmę viename arba kitame lauke, t.y. asmuo gali būti arba darbuotojas, arba vadybininkas, tačiau jis negali užimti abiejų pareigų, kaip jokios pareigos. Atributo reikšmės apribojimas šiems laukams atrodo taip:

```
ALTER TABLE [dbo].[Person]
ADD CONSTRAINT [CK_Person_2]
CHECK (([employeeID] IS NOT NULL AND [managerID] IS NULL
OR [employeeID] IS NULL AND [managerID] IS NOT NULL))
```

Tačiau visus šiuos apribojimus perkėlus į taikomąją programą, kurie suveiktų dar nespėjus nusiųsti užklausos į duomenų bazę, nekorektiškai suvestų duomenų atveju programa nutrauktų operaciją, vietoj to kad būtų laukiama kol duomenų bazė grąžins klaidos požymį, jog reikšmė neatitinka sąlygų.

Tiriamoje sistemoje šis apribojimas lentelės PERSON laukui *sex* realizuotas taikomojoje programoje taip, kad vartotojas net neturi galimybės įvesti nekorektišką reikšmę. Programos lange iš iškrentančio meniu leidžiama pasirinkti tik „Male“ arba „Female“ reikšmes.

PERSON lentelės laukams *employeeID* ir *managerID* taip pat realizuotas šis apribojimas, leidžiant programoje vartotojui pasirinkti tik vieną iš šių dviejų laukų, o pamiršus įvesti į kodo laukelį kokią nors reikšmę, akimirksniu parodomas pranešimas ir įterpimo užklausa net nesiunčiama į duomenų bazę.

3.7.7. Išvestinio atributo apribojimas

Šio apribojimo principas yra toks, kad lauko reikšmė gaunama pagal kitų laukų reikšmių apskaičiavimus ar reikšmių apdorojimus. Išvestinio atributo apribojimas duomenų bazėje dažniausiai realizuojamas trigeriu, tačiau tai galima užtikrinti ir funkcija taikomosios programos lygyje. Pastarasis būdas bus šiek tiek lėtesnis, kadangi reikalingi du kreipiniai į duomenų bazę: vienas – išvedant apskaičiuojamą reikšmę, kitas – įrašant šią reikšmę į tam tikrą lentelės lauką. Taigi remiantis šio tiriamojo darbo tikslu, rasti apribojimų strategijos realizaciją, kuri užtikrintų duomenų kokybiškumą, bet tuo pačiu ir veiktų optimaliu režimu, nesulėtinant kitų procesų ir pan., tokie apribojimai paliekami duomenų bazės lygyje.

Pasirinktos dalykinės srities atveju, tokį išvestinį apribojimą galima taikyti PRODUCT lentelės laukui *price*. Tarkime kuriant naują produkto operacijos įrašą lentelėje PRODUCTOPERATION, reikalinga, kad pagal parinkto produkto operacijos tipo trukmę ir valandinį tarifą būtų paskaičiuojama pardavimo kaina su 35% antkainiu. Tokiu atveju lentelei PRODUCTOPERATION kuriamas trigeris, kurio programinis kodas atrodytų taip:

```
CREATE TRIGGER [PRODUCT_PRICE] ON [DBO].[PRODUCTOPERATION]
FOR INSERT, UPDATE
AS
DECLARE @PRICE INT, @CODE VARCHAR(20)

SELECT @CODE = ISNULL(P.CODE,''),
        @PRICE = ISNULL(SUM(O.[TIME]*O.TARIFF*1.35),0)
FROM INSERTED INS
INNER JOIN PRODUCTOPERATION P ON P.CODE = INS.CODE
INNER JOIN OPERATIONTYPE O ON O.OPERATIONTYPEID = P.OPERATIONTYPEID
GROUP BY P.CODE

UPDATE PRODUCT SET PRICE = @PRICE WHERE CODE = @CODE
```

Išvestinio atributo apribojimo reikėtų ir PERSON lentelės laukui *age* apskaičiavimui pagal įvestą gimimo datą. Šiuo atveju apribojimą galima užtikrinti ir taikomojoje programoje sukurta funkcija *get_age(<gimimo_data>)*. Joje apskaičiuojamas ir grąžinamas skirtumas tarp einamosios ir gimimo datos metais. Šios funkcijos kodas MS FoxPro sintakse gana nesudėtingas:

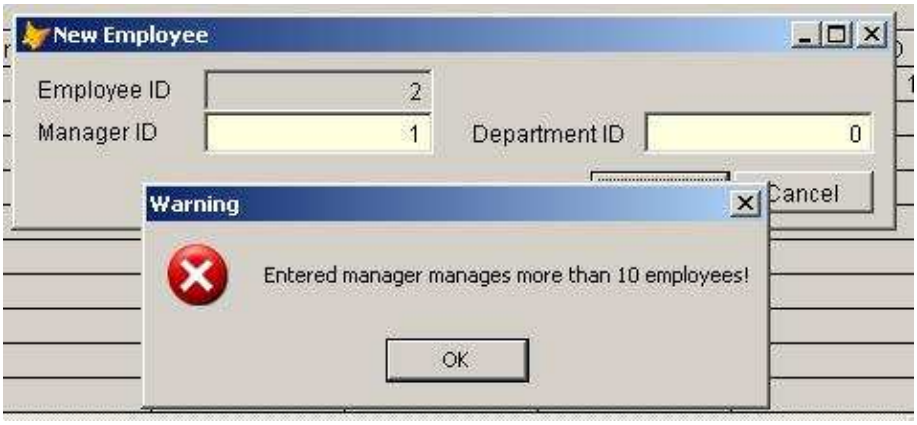
```
FUNCTION get_age
LPARAMETERS birthday
RETURN ROUND((DATE() - birthday)/365, 0)
```

Žinoma tokie elementarūs skaičiavimai sparčiai veikia tiek duomenų bazės, tiek ir taikomosios programos lygyje, dėl to nustatyti, kuris būdas yra optimalus, yra gana sudėtinga, Klaidų tikimybės abiem atvejais yra pakankamai mažos, o apribojimo veikimo trukmių skirtumas praktiškai nepastebimas. Tokiu atveju optimalų veikimą labiau lemia kreipinių i duomenų bazę, papildomų operacijų skaičiai. Tačiau iš praktikos pastebėta, kad jeigu reikia apskaičiuoti reikšmę pagal sudėtingas finansines ar kitokias formules, o tuo pačiu metu duombazėje vykdoma dar šimtai kitų užklausų, skaičiavimai pastebimai greičiau atliekami kliento kompiuteryje.

3.7.8. Kardinalumo apribojimas

Kardinalumo apribojimas taip pat dažniausiai realizuojamas trigeriais. Jo esmė yra apriboti lentelės įrašų kiekį, susijusių su kitos susietos lentelės vienu įrašu. Šioje dalykinėje srityje toks apribojimas taikomas EMPLOYEE lentelei, tačiau ne trigeriu, o taikomosios programos funkcija *get_count(<lauko_reikšmė>)*. Šis apribojimo realizacijos būdas pasirinktas dėl tų pačių priežasčių, kaip ir išorinio atributo unikalumo apribojimo atveju – jeigu kardinalumo tikrinimas grąžina reikšmę, kuri neatitinka apribojimo reikalavimų, tuomet atliekama viena duomenų bazės užklausa mažiau, nemėginant terpt naujo įrašo į ją.

Pasirinktos dalykinės srities atveju darbuotojui negalima priskirti vadybininko, jeigu pastarasis jau vadovauja daugiau negu dešimčiai darbuotojų. Taigi, jeigu iškviesta funkcija *get_manager_count(managerID)* gražins skaičių didesnį nei 10, programa parodys pranešimą ir operaciją nutrauks:



11 pav. Daugialypiškumo apribojimas

Funkcijos veikimo principas pagrįstas tuo, kad pagal jai paduotus parametrus (lentelės pavadinimą, lauko pavadinimą ir lauko reikšmę), ji grąžina skaičių lygų įrašų pasikartojimui su tokia reikšme. Funkcijos kodas MS FoxPro sintakse atrodo taip:

```
FUNCTION get_manager_count
LPARAMETERS val_
SQLEXEC(hConn,“SELECT COUNT(*) AS c FROM EMPLOYEE WHERE MANAGERID = ?val_”, ”count”)
RETURN count.c
```

Kardinalumo apribojimo reikia ir lentelei TASK. Juo turėtų būti draudžiama priskirti naują užduotį darbuotojui, jeigu jo vienos dienos užduočių suminė trukmė viršija 8 valandas. Kuriant naują užduotį, dėl išorinio rakto apribojimų, reikalinga siųsti tris įterpimo užklausas į skirtingas lenteles: PRODUCTIONTASK (arba PLANNEDTASK), TASK ir TASKRELATION. Į lentelę TASK įterpinėja antroji užklausa, dėl to, jeigu naudosime trigerinį apribojimą, kuris aptiks nekorektiškus duomenis, reiks atšaukti ne tik šią, bet ir pirmąją užklausą. Taip sugaištama palyginus nemažai laiko. Praktiškiau būtų įrašų korektiškumą patikrinti taikomojoje programoje prieš siunčiant įterpimo užklausas.

Tiriamos sistemos atveju, šis apribojimas realizuotas taikomosios programos funkcija *check_performer(performerID, taskDate, seqNumber, code)*. Jos programinis kodas pateiktas apačioje:

```
FUNCTION check_performer
LPARAMETERS perf_,data_, seq_number_, code_
LOCAL sql_
sql_="select sum(trukme) trukme from ( "+;
```

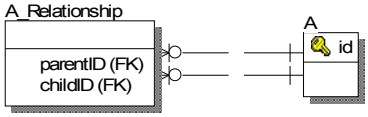
```
 "select isnull(sum([time]),0) trukme from task t  "+;
 "inner join productoperation p on p.seqnumber = t.seqnumber and p.code = t.code "+;
 "inner join operationtype o on o.operationtypeid = p.operationtypeid "+;
 "inner join productiontask pr on pr.productiontaskid = t.productiontaskid  "+;
 "where t.performer = ?perf_ and pr.fromproductiondate = ?data_ "+;
 "union all  "+;
 "select isnull(sum([time]),0) trukme from task t  "+;
 "inner join productoperation p on p.seqnumber = t.seqnumber and p.code = t.code "+;
 "inner join operationtype o on o.operationtypeid = p.operationtypeid "+;
 "inner join plannedtask pl on pl.plantaskid = t.plantaskid  "+;
 "where t.performer = ?perf_ and pl.fromplanneddate = ?data_ ) a "
SQLEXEC(hConn,sql_,"normat")

sql_="select [time] trukme from productoperation p "+;
 "inner join operationtype o on o.operationtypeid = p.operationtypeid "+;
 "where p.seqnumber = ?seq_number_ and p.code = ?code_
SQLEXEC(hConn,sql_,"new_task")

IF normat.trukme+new_task.trukme > 100
  RETURN .F.
ENDIF
RETURN .T.
```

3.7.9. Refleksyvių ryšių apribojimai

Tokie apribojimai duomenų bazėje taikomi ryšiams, kaip pavaizduota paveikslėlyje:



12 pav. Refleksyvus ryšys

Refleksyvių ryšių apribojimas naudojamas norint užtikrinti visų *parentID* ir *childID* kombinacijų nepriekaištingumą. Mūsų duomenų bazėje toks ryšys taipogi yra tarp lentelių TASK ir TASKRELATION. Pastarojoje lentelėje saugoma užduočių seka produktui sukurti, kadangi gamybos procesas turi vykti nuosekliai. Taigi šios lentelės įrašams taikysime asimetrinį refleksyvių

ryšių apribojimą, kuris neleidžia atsirasti įrašams su priešinga veiksmų seką, t.y. jeigu veiksmų kryptis yra 1 → 2, tai turėtų neleisti įterpti įrašą su seka 2 → 1. Šis apribojimas realizuojamas trigeriu, nes taikomojoje programoje jis būtų per daug painus ir sudėtingas. Trigerio SQL sintaksė pateikta žemiau:

```
CREATE TRIGGER [dbo].[asymmetric] ON [TaskRelation]
FOR INSERT, UPDATE
AS
BEGIN
  DECLARE @id INT, @afterTaskID INT, @beforeTaskID INT, @count INT
  DECLARE seq CURSOR FOR
  SELECT [id], afterTaskID, beforeTaskID FROM taskrelation
  OPEN seq
  WHILE 1=1
  BEGIN
        FETCH NEXT FROM jp_prek INTO @id, @afterTaskID, @beforeTaskID
        IF    @@FETCH_STATUS <> 0 BREAK

        SELECT @count=count(*) FROM taskrelation
        WHERE afterTaskID = @beforeTaskID and beforeTaskID = @afterTaskID

        IF @count>0
        BEGIN
              RAISERROR('Invalid record in table TaskRelation',16,1)
              ROLLBACK TRANSACTION
        END
  END
  CLOSE seq
  DEALLOCATE seq
END
```

Galima naudoti ir antisimetrinį („Antysymmetric“) apribojimą. Jo veikimo principas panašus kaip ir asimetrinio apribojimo, tik jis leidžia tai pačiai reikšmei dalyvauti abejose apribojimo rolėse. Taigi šių dviejų apribojimų realizavimas yra labai panašus, tik antisimetriniu atveju į tikrinimą įtraukiama papildoma sąlyga, kuri neiššaukia klaidos esant vienodoms laukų reikšmėms.

Tačiau turint tokį refleksyvų ryšį, reikalingas ir korektiškas įrašų šalinimas iš lentelės, nesugriaunant jų sekos. Pirmiausia šalinamas įrašas, kurio *afterTaskID* reikšmė sutampa su šalinamos užduoties numeriu. Tuomet suveikia trigeris, kuris sutvarko likusius ryšius lentelėje. Šį apribojimą realizuosime taip pat trigeriu, kuris SQL sintakse atrodytų taip:

```
CREATE TRIGGER [dbo].[fix_relations] ON  [dbo].[TaskRelation]
FOR DELETE
AS
BEGIN
   DECLARE @parent INT, @v INT
  SELECT @parent = beforeTaskID, @child = afterTaskID
  FROM DELETED
  IF @parent <> @child
        UPDATE taskrelation SET beforeTaskID = @parent WHERE beforeTaskID = @child
  ELSE
        UPDATE taskrelation SET beforeTaskID = afterTaskID
        WHERE beforeTaskID = @child AND beforeTaskID <> afterTaskID
END
```

4. Eksperimentinis apribojimų realizacijos būdų tyrimas

Norint nustatyti, kokiu būdu efektyviausiai realizuojami duomenų vientisumo apribojimų tipai buvo vykdomas eksperimentas. Pagal jo metu gautus išmatuotus kriterijus galima bus sudaryti apribojimų realizavimo strategijos parinkimo metodiką bei rekomendacijas.

Eksperimento vykdymui pasirinktas didelės įmonės serveris, kurio charakteristika:

- 2 Dual Core 2,8 MGz procesoriai;
- 8 GB operatyvinė atmintis;
- Windows 2000 Server operacinė sistema;
- MS SQL Server 2000 duomenų bazė;
- ~2000 į tinklą sujungtų kompiuterių.

Duomenų vientisumo apribojimams realizuoti ir tirti pasirinkta gamybos valdymo sistema, su gamybos dalyviais, jų vykdomomis užduotimis, pagamintos produkcijos apskaita. Buvo sukurta atskira eksperimentinė duomenų bazė pagal schemą pateiktą 8 pav. Norint, kad tiriamos sistemos elgsena būtų kuo realistiškesnė, sukurta 500 susijungimų su naujai sukurta duomenų baze, kur per kiekvieną iš jų ciklišk ai siunčiamos užklausos į duomenų bazę traukiant duomenis iš įvairių lentelių, taip apkraunant duomenų bazę, lyg ja naudotųsi šimtai vartotojų vienu metu. Taip pat kiekvienai lentelei sugeneruoti skirtingai kiekiai duomenų, nuo 5 000 iki 10 000 000.

Norint statistiškai ištirti apribojimų veikimo charakteristikas, šiuo atveju – veikimo trukmę, reikia paskaičiuoti imties vidurkį, vidutinį kvadratinį nuokrypį, vidurkio patikimumo intervalą. Šių parametrų apskaičiavimui, naudojamos žemiau pateiktos formulės:

- Imties vidurkis:

$$\bar{x} = \frac{\sum x_i}{n} \qquad (1)$$

n – imties dydis

- Imties kvadratinis nuokrypis:

$$s = \sqrt{\frac{\sum (x_i - \bar{x})^2}{n-1}} \qquad (2)$$

- Variacijos koeficientas:

$$CV = \frac{s}{\bar{x}} \cdot 100\% \qquad (3)$$

- Standartinė imties paklaida:

$$S = \frac{s}{\sqrt{n}} \qquad (4)$$

Taip pat apskaičiuojamas vidurkio 95% patikimumo intervalas pagal Stjudento pasiskirstymą. Laisvės laipsnių skaičius mūsų imties atveju yra *df =30 – 1 = 29*. Stjudento pasiskirstymų lentelės 0,025 stulpelyje 29 eilutėje randame reikšmę *t = 2,04*.

Patikimumo intervalo skaičiavimo formulė:

$$\bar{x} - t\frac{s}{\sqrt{n}} \le \mu \le \bar{x} + t\frac{s}{\sqrt{n}} \qquad (5)$$

4.1. Išorinio unikalumo apribojimo tyrimas

Atliktas eksperimentinis tyrimas, realizuojant išorinio unikalumo apribojimą lentelėms COUNTRY ir PERSON. Pirmąjį eksperimentą atliksime su COUNTRY lentele. Joje yra ~10 000 įrašų. Šis apribojimas, keičiant PartOfPersonCode reikšmę, tikrina ar nėra kitų šalių su tokiu kodu, kuriose būtų asmenų su tokiais asmens kodais, kaip ir redaguojamos šalies.

Imčiai atsitiktinai atrinkta 30 reikšmių, neatsižvelgiant į tai, ar jos tenkina apribojimo sąlygas, ar ne. Apribojimo veikimo trukmės kiekvienu imties elemento atveju fiksuojamos milisekundės tikslumu. Tokio detalumo mato užtenka, norint pastebėti skirtumus tarp dviejų apribojimo realizacijos būdų – programuojamo duomenų bazės trigeriu (toliau DB) ir programuojamu taikomojoje programoje (toliau TP).

Veikimo trukmės milisekundėmis kiekvienu imties elemento atveju pateiktos žemiau:

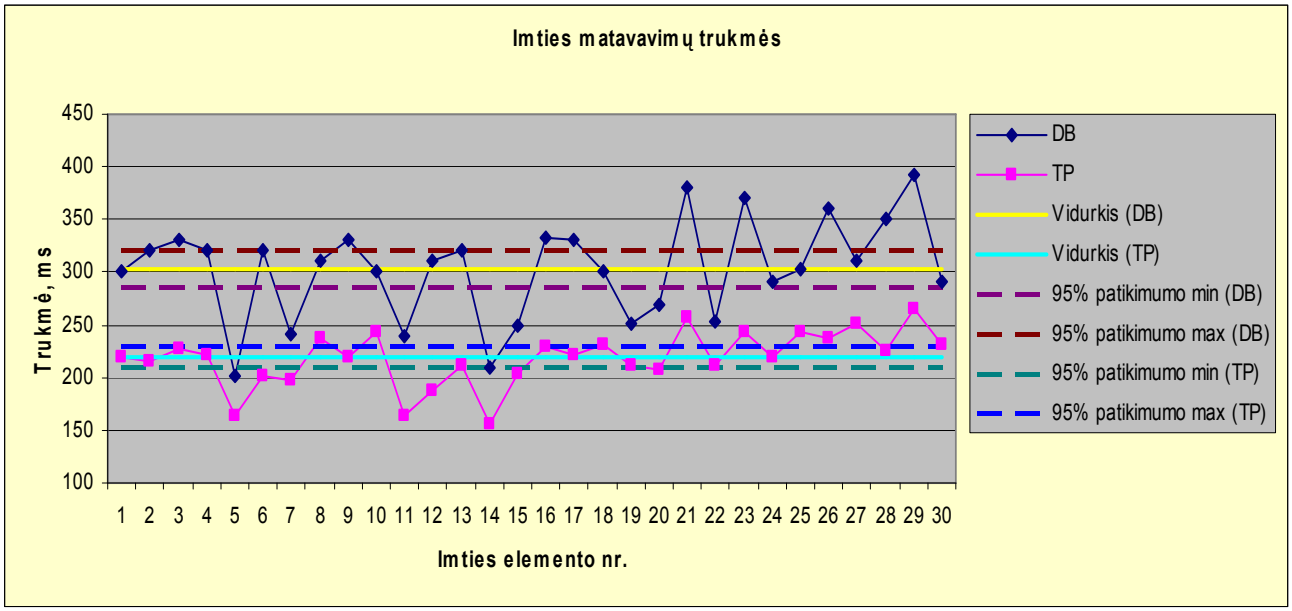
Nr.	DB	TP
1	301	220
2	321	215
3	330	227
4	321	222
5	202	164
6	320	201
7	241	198
8	311	237
9	330	220
10	301	244
11	240	164
12	311	187
13	320	211
14	210	156
15	250	204

Nr.	DB	TP
16	333	230
17	331	221
18	301	231
19	251	212
20	270	208
21	380	258
22	254	211
23	371	243
24	291	220
25	302	243
26	361	237
27	311	251
28	351	225
29	393	266
30	290	231

Pagal šias imtis apskaičiuoti parametrai pateikti 1 lentelėje. Grafiškai jie atvaizduoti 13 paveikslėlyje.

1 lentelė. Imties parametrai lentelei COUNTRY

	Imties vidurkis	Kvadratinis nuokrypis	Variacijos koef., %	Standartinė paklaida	Vidurkio 95% patikimumo intervalas (t_{29}=2,04)
DB	303,3	47,3	15,6	8,6	285,7 ÷ 320,9
TP	218,6	26,3	12	4,8	208,8 ÷ 228,4



13 pav. Lentelės COUNTRY išorinio unikalumo apribojimo charakteristika

Iš eksperimento rezultatų nesunkiai pastebima, kad apribojimą realizavus taikomojoje programoje, vidutiniškai jo veikimo trukmė buvo trumpesnė beveik 100 ms. Staigūs ir dideli kreivių šuoliai į apačią žymi apribojimo nekorektiškų duomenų atmetimo momentus, kai įrašas nebuvo įkeliamas į lentelę.

Kitas išorinio unikalumo apribojimas realizuotas lentelėje PERSON. Jo principas yra toks, kad negali būti dviejų asmenų su vienodais asmens kodais ir šalių PartOfPersonCode kodais. Šioje lentelėje ~30 000 įrašų.

Imčiai atsitiktinai atrinkta 30 reikšmių, neatsižvelgiant į tai, ar jos tenkina apribojimo sąlygas, ar ne. Kiekvieno imties elemento atveju užfiksuotos apribojimo veikimo trukmės milisekundėmis:

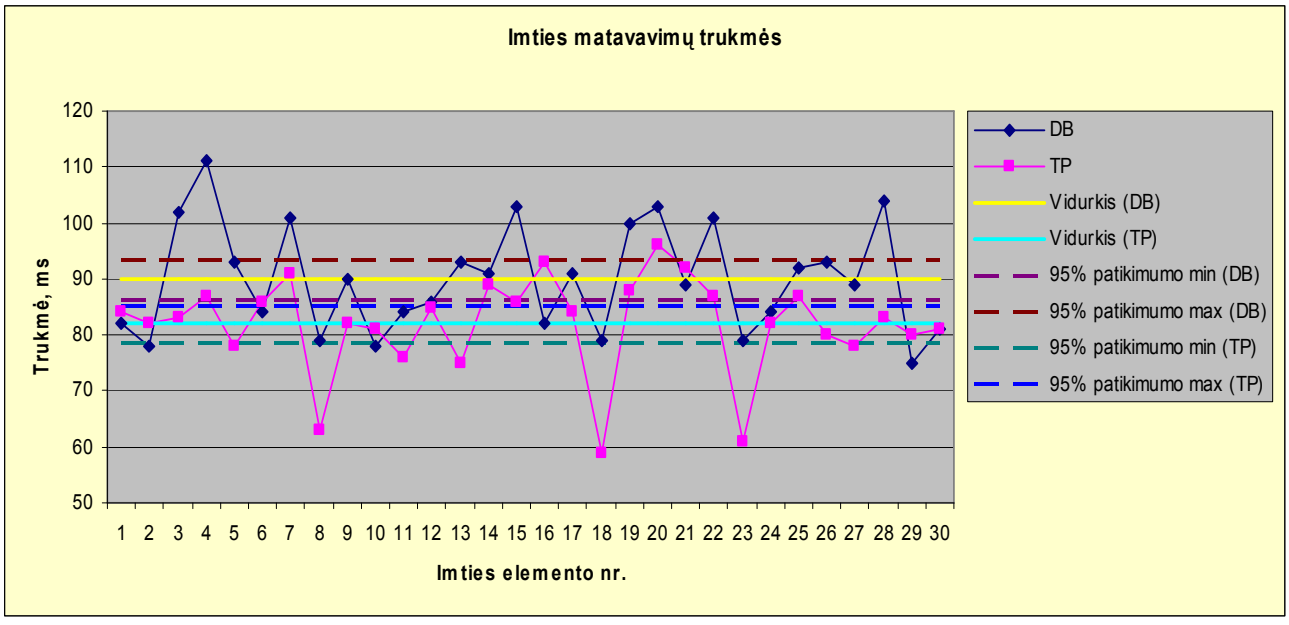
Nr.	DB	TP
1	82	84
2	78	82
3	102	83
4	111	87
5	93	78
6	84	86
7	101	91
8	79	63
9	90	82
10	78	81
11	84	76
12	86	85
13	93	75
14	91	89
15	103	86

Nr.	DB	TP
16	82	93
17	91	84
18	79	59
19	100	88
20	103	96
21	89	92
22	101	87
23	79	61
24	84	82
25	92	87
26	93	80
27	89	78
28	104	83
29	75	80
30	81	81

Pagal šias imtis apskaičiuoti parametrai pateikti 3 lentelėje. Grafiškai jie atvaizduoti 14 paveikslėlyje.

3 lentelė. Imties parametrai lentelei PERSON

	Imties vidurkis	Kvadratinis nuokrypis	Variacijos koef., %	Standartinė paklaida	Vidurkio 95% patikimumo intervalas (t_{29}=2,04)
DB	89,9	9,7	10,7	1,8	86,3 ÷ 93,5
TP	82,0	8,6	10,5	1,6	78,7 ÷ 85,2



14 pav. Lentelės PERSON išorinio unikalumo apribojimo charakteristika

Iš grafiko matyti, kad ir PERSON lentelės išorinio unikalumo apribojimą realizavus apribojimų valdiklyje, operacijos vykdomos 7-10 ms greičiau, nei realizavus trigeryje.

4.2. Kardinalumo apribojimo tyrimas

Kardinalumo apribojimo tyrimas pradedamas nuo EMPLOYEE lentelės. Lentelėje apribojimas turi sustabdyti įrašymo arba atnaujinimo operaciją, jeigu įrašomas vadybininko kodas yra priskirtas 10-čiai darbuotojų. Joje iš viso yra ~5000 įrašų.

Imčiai atsitiktinai atrinkta 30 reikšmių, neatsižvelgiant į tai, ar jos tenkina apribojimo sąlygas, ar ne. Kiekvieno imties elemento atveju užfiksuotos apribojimo veikimo trukmės milisekundėmis:

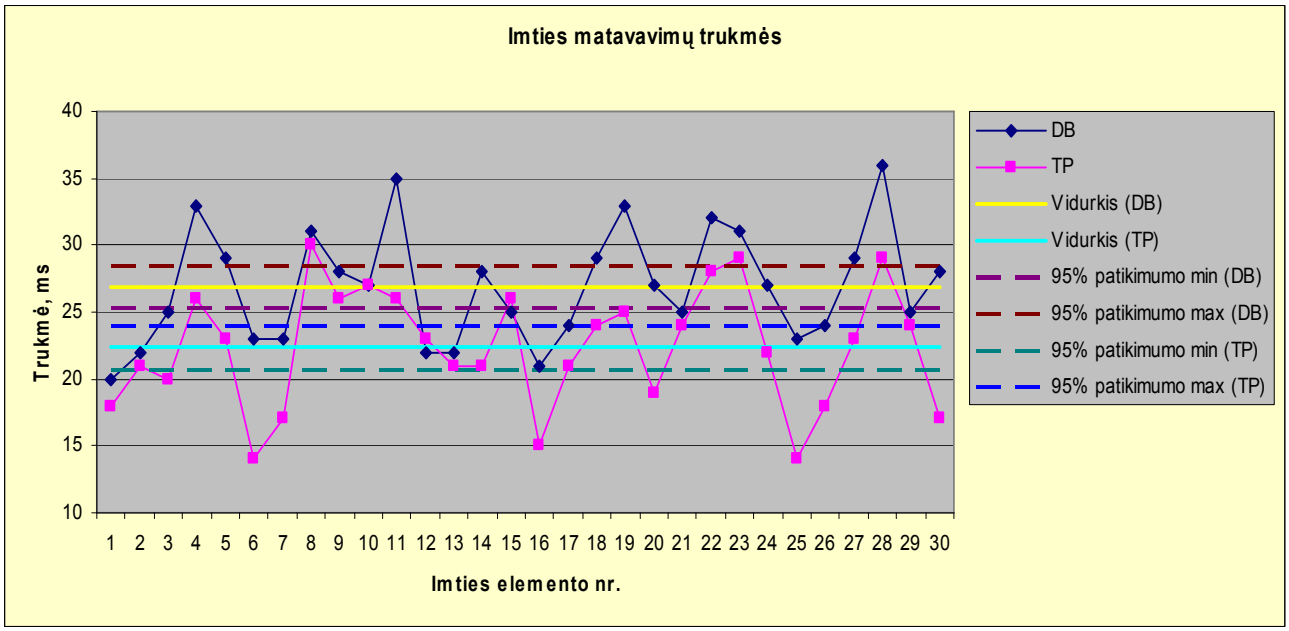
Nr.	DB	TP
1	20	18
2	22	21
3	25	20
4	33	26
5	29	23
6	23	14
7	23	17
8	31	30
9	28	26
10	27	27
11	35	26
12	22	23
13	22	21
14	28	21
15	25	26

Nr.	DB	TP
16	21	15
17	24	21
18	29	24
19	33	25
20	27	19
21	25	24
22	32	28
23	31	29
24	27	22
25	23	14
26	24	18
27	29	23
28	36	29
29	25	24
30	28	17

Pagal šias imtis apskaičiuoti parametrai pateikti 4 lentelėje. Grafiškai jie atvaizduoti 15 paveikslėlyje.

4 lentelė. Imties parametrai lentelei EMPLOYEE

	Imties vidurkis	Kvadratinis nuokrypis	Variacijos koef., %	Standartinė paklaida	Vidurkio 95% patikimumo intervalas (t_{29}=2,04)
DB	26,9	4,3	15,9	0,8	25,3 ÷ 28,5
TP	22,4	4,5	20	0,8	20,7 ÷ 24



15 pav. Lentelės EMPLOYEE kardinalumo apribojimo charakteristika

Šiuo eksperimento atveju išmatuotos operacijų trukmės buvo ganėtinai mažos, dėl to ir skirtumas tarp abiejų apribojimų realizavimo būdų yra nedidelis, vos kelios milisekundės, tačiau ir čia pranašesnis pasirodė apribojimų realizavimas apribojimų valdiklyje.

Sekantis tyrimo objektas, kuriame naudojamas kardinalumo apribojimas – lentelė TASK. Šioje lentelėje yra didžiausias kiekis įrašų ~10 000 000, o vienai užduočiai sukurti, siunčiamos net 3 įterpimo užklausos. Dėl to eksperimento metu gautos operacijų trukmės yra kur kas ilgesnės už ankstesniuose eksperimentuose išmatuotas trukmes.

Apribojimas neleidžia priskirti užduoties darbuotojui, kurio dienos užduočių suminė trukmė viršyja 8 valandas. Tiriamoje sistemoje šios lentelės apribojimo veikimas trunka gana ilgai ir dėl to, kad norint patikrinti apribojimo sąlygą, reikia duomenis nuskaitinėti net iš penkių lentelių.

Imčiai atsitiktinai atrinkta 30 reikšmių, neatsižvelgiant į tai, ar jos tenkina apribojimo sąlygas, ar ne. Kiekvieno imties elemento atveju užfiksuotos apribojimo veikimo trukmės milisekundėmis:

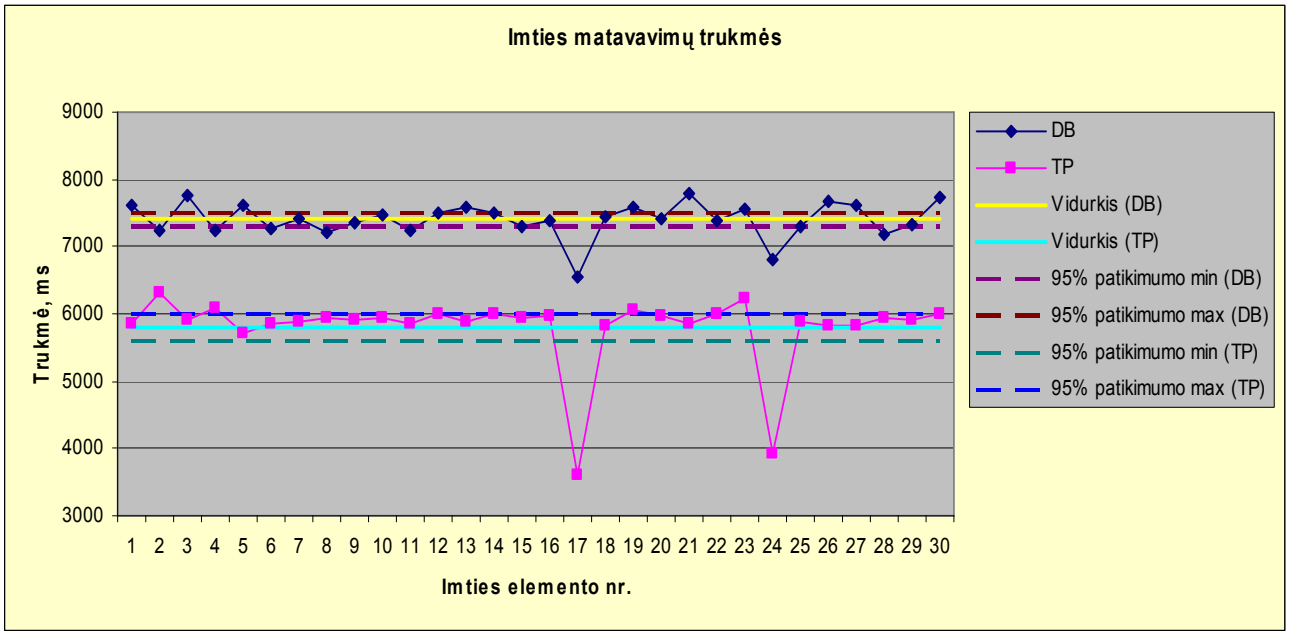
Nr.	DB	TP
1	7614	5855
2	7233	6306
3	7764	5906
4	7253	6086
5	7614	5715
6	7283	5866
7	7414	5886
8	7203	5946
9	7343	5917
10	7483	5938
11	7253	5865
12	7494	5996
13	7574	5895
14	7504	6005
15	7294	5936

Nr.	DB	TP
16	7373	5966
17	6535	3605
18	7453	5815
19	7593	6046
20	7403	5975
21	7784	5865
22	7393	6006
23	7554	6226
24	6795	3936
25	7284	5876
26	7684	5835
27	7624	5815
28	7193	5946
29	7333	5905
30	7724	6001

Pagal šias imtis apskaičiuoti parametrai pateikti 5 lentelėje. Grafiškai jie atvaizduoti 16 paveikslėlyje.

5 lentelė. Imties parametrai lentelei TASK

	Imties vidurkis	Kvadratinis nuokrypis	Variacijos koef., %	Standartinė paklaida	Vidurkio 95% patikimumo intervalas (t_{29}=2,04)
DB	7401,6	265,4	3,6	48,4	7302,8 ÷ 7500,4
TP	5795,9	565	9,7	103,1	5587,4 ÷ 6008,3



16 pav. Lentelės TASK kardinalumo apribojimo charakteristika

4.3. Išvestinės reikšmės apribojimo tyrimas

Išvestinės reikšmės apribojimo tyrimas atliekamas lentelei PRODUCTOPERATION. Šis apribojimas neaptinka nekorektiškų duomenų ir neatšaukia jų įterpimo ar atnaujinimo, jo tikslas – pagal įterpinėjamus įrašus apskaičiuoti kitos lentelės lauko reikšmę, dėl to realizuodami jį taikomosios programos lygyje, turime bet kuriuo atveju siūsti i duomenų bazę dvi užklausas, vieną – duomenų ištraukimui naujos reikšmės apskaičiavimui, kitą – apskaičiuotų duomenų įterpimui. PRODUCTOPERATION lentelėje yra ~5 000 000 įrašų.

Imčiai sugeneruota 30 atsitiktinių reikšmių. Kiekvienos atveju užfiksuotos apribojimo veikimo trukmės milisekundėmis:

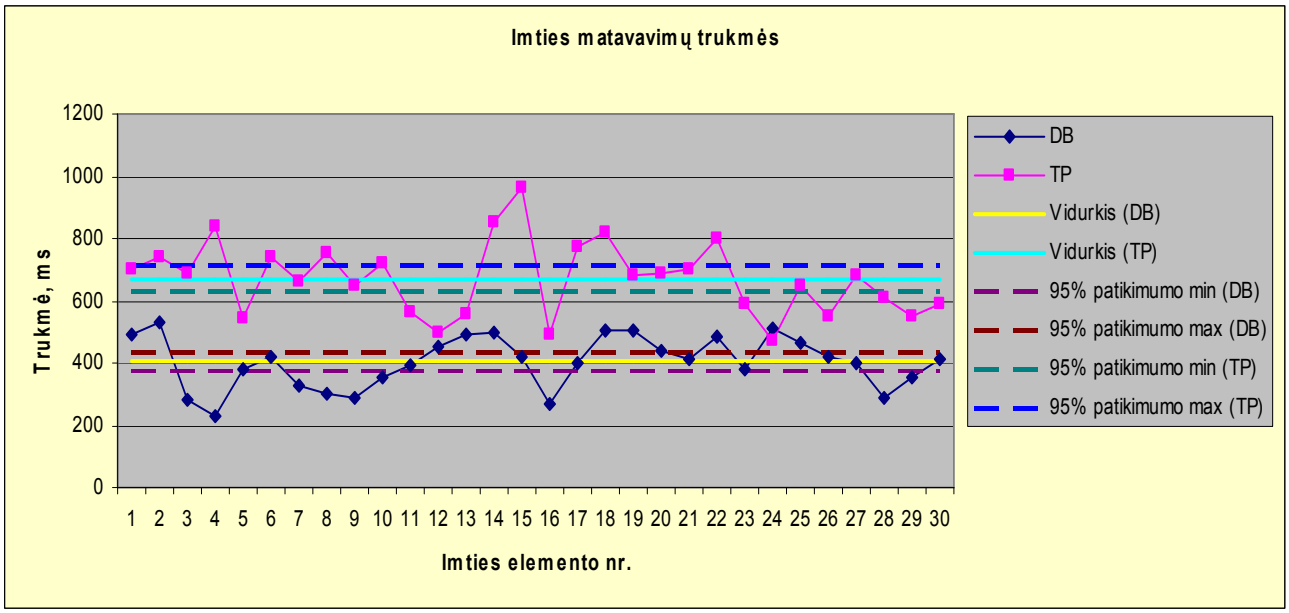
Nr.	DB	TP
1	492	701
2	531	743
3	280	691
4	231	841
5	381	545
6	422	742
7	331	661
8	301	753
9	290	651
10	356	721
11	392	564
12	451	501
13	489	560
14	501	852
15	421	961

Nr.	DB	TP
16	271	491
17	398	771
18	502	820
19	504	681
20	438	691
21	412	702
22	483	802
23	380	591
24	512	470
25	467	651
26	421	550
27	399	681
28	289	611
29	357	551
30	416	590

Pagal šias imtis apskaičiuoti parametrai pateikti 6 lentelėje. Grafiškai jie atvaizduoti 17 paveikslėlyje.

6 lentelė. Imties parametrai lentelei PRODUCTOPERATION

	Imties vidurkis	Kvadratinis nuokrypis	Variacijos koef., %	Standartinė paklaida	Vidurkio 95% patikimumo intervalas (t29=2,04)
DB	403,9	82,5	20,4	15,1	373,2 ÷ 434,7
TP	671,3	118,2	17,6	21,6	627,3 ÷ 715,4



17 pav. Lentelės PRODUCTOPERATION išvestinės reikšmės apribojimo charakteristika

Skirtingai nei eksperimentuojant su kitais duomenų vientisumo apribojimų tipais, pastarojo veikimo trukmė, kaip matyti iš grafiko, yra kur kas trumpesnė realizuojant apribojimą trigeriu, nei apribojimų valdikliu.

4.4. Kombinuotų apribojimų tyrimas

Žinant, kad duomenų vientisumo apribojimų tipų realizuotų skirtingais būdais veikimo trukmės yra skirtingos, t.y. vieni apribojimų tipai veikia sparčiau realizavus juos trigeriu, kiti – apribojimų valdikliu, galima atlikti eksperimentą parinkus tam tikrą lentelės apribojimų kombinaciją ir ją realizavus skirtingais duomenų vientisumo apribojimų realizavimo būdais. Šiam eksperimentui parinkta lentelė TASK su trimis duomenų vientisumo apribojimų tipais: kardinalumo, išvestinės reikšmės ir refleksyvinių ryšių.

Iš ankstesnių tyrimų rezultatų pastebėta, kad kardinalumo apribojimas veikia sparčiau, jį realizavus apribojimų valdikliu, tuo tarpu išvestinės reikšmės apribojimas – realizuojant trigeriu. Eksperimentuojant tiriami trys duomenų vientisumo apribojimų realizavimo būdai. Pirmu atveju bus tiriamos veikimo charakteristikos, kai šie abu duomenų vientisumo apribojimų tipai realizuoti trigeriais, antru atveju jie bus realizuoti taikomojoje programoje, o paskutiniu atveju – atsižvelgiant į tai, kokiu realizacijos būdu apribojimas veikia sparčiausiai, t.y. kardinalumo apribojimas – apribojimų valdikliu, išvestinės reikšmės – trigeriu. Trečiasis duomenų vientisumo apribojimo tipas, refleksyvinių ryšių apribojimas, visais atvejais bus realizuotas trigeriu, kadangi apribojimų valdiklyje jo veikimas būtų labai sudėtingas ir painus.

Imčiai sugeneruota 30 atsitiktinių reikšmių. Kiekvienos atveju užfiksuotos apribojimo veikimo trukmės milisekundėmis:

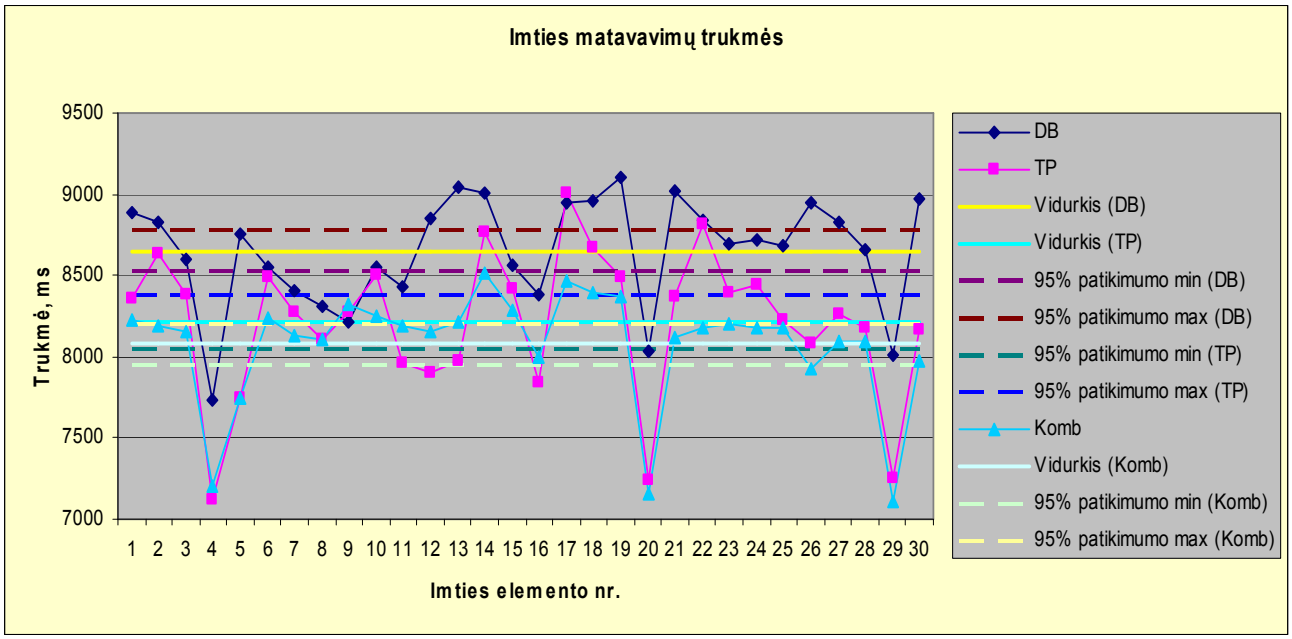
Nr.	DB	TP	Komb.
1	8890	8358	8226
2	8826	8635	8184
3	8604	8379	8154
4	7729	7118	7207
5	8757	7750	7743
6	8549	8492	8235
7	8407	8269	8131
8	8306	8105	8104
9	8213	8270	8327
10	8551	8501	8251
11	8429	7957	8185
12	8847	7899	8151
13	9041	7975	8209
14	9007	8761	8515
15	8557	8419	8281

Nr.	DB	TP	Komb.
16	8386	7839	7992
17	8946	9009	8472
18	8959	8675	8391
19	9105	8489	8373
20	8031	7246	7161
21	9020	8371	8122
22	8842	8812	8182
23	8694	8399	8204
24	8717	8448	8179
25	8685	8229	8173
26	8947	8085	7923
27	8821	8258	8095
28	8660	8179	8098
29	8004	7258	7112
30	8972	8171	7970

Pagal šias imtis apskaičiuoti parametrai pateikti 7 lentelėje. Grafiškai jie atvaizduoti 18 paveikslėlyje.

7 lentelė. Imties parametrai esant kombinuotiems apribojimų būdams lentelei TASK

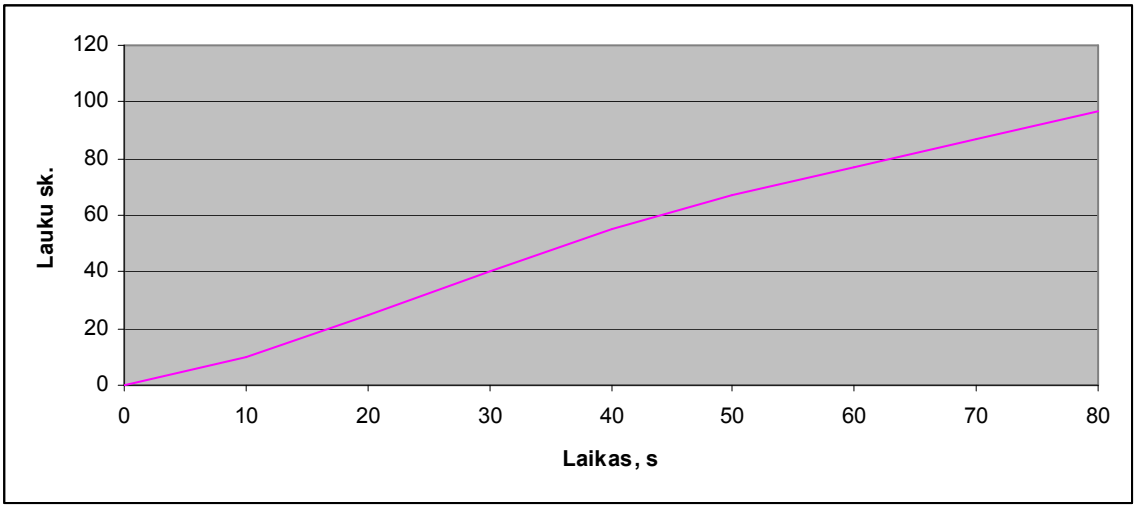
	Imties vidurkis	Kvadratinis nuokrypis	Variacijos koef., %	Standartinė paklaida	Vidurkio 95% patikimumo intervalas (t29=2,04)
DB	8650,1	341.3	3.9	62.3	8522.9 ÷ 8777.2
TP	8211.9	446.7	5.4	81.6	8045.5 ÷ 8378.3
Komb.	8078.3	347.5	4.3	63.4	7948.9 ÷ 8207.8



18 pav. Lentelės TASK kombinuotų apribojimų charakteristika

4.5. Užklausų optimizavimo tyrimas

Kitas svarbus veikimo pagerinimo faktorius yra užklausų veikimas, todėl duomenų bazių kūrėjai stengiasi rasti optimalią duomenų bazės užklausų struktūrą. Akivaizdu, kad kuo didesnis lentelės laukų ir įrašų skaičius, tuo ilgiau bus vykdoma užklausa. Dėl to tyrimui buvo pasirinkta lentelė, kurioje yra arti pusės milijono įrašų, o laukų skaičius siekia 97. Kadangi užklausos sakinys pradedamas iš lentelės norimų grąžinamų laukų aprašymu, nuo jo ir pradėta analizė. Priklausomai nuo grąžinamų laukų kiekio, ženkliai skiriasi užklausos vykdymo laikai. Jie atsispindi pateiktame grafike.



19 pav. Užklausos reakcijos laiko priklausomybė nuo užklausoje užduotų laukų skaičiaus

Suprantama, kad didinant laukų kiekį, žinoma ilgėja ir veikimo laikas, tačiau norint rezultate turėti visus lentelės laukus, juos paeiliui išvardinant, rezultatai grąžinami greičiau, nei nevardinus laukų, o tik nurodžius žvaigždutės „*“ simbolį. Skirtumas tarp šių rezultatų analizuojamu atveju yra apie 7 sekundės. Taigi galima daryti išvadą, kad vykdant užklausą, geriausiai būtų nurodyti tik tuos laukus, kurie tikrai reikalingi, o ne išsivesti visus rezultatus ir tik tada atsirinkti ko reikia.

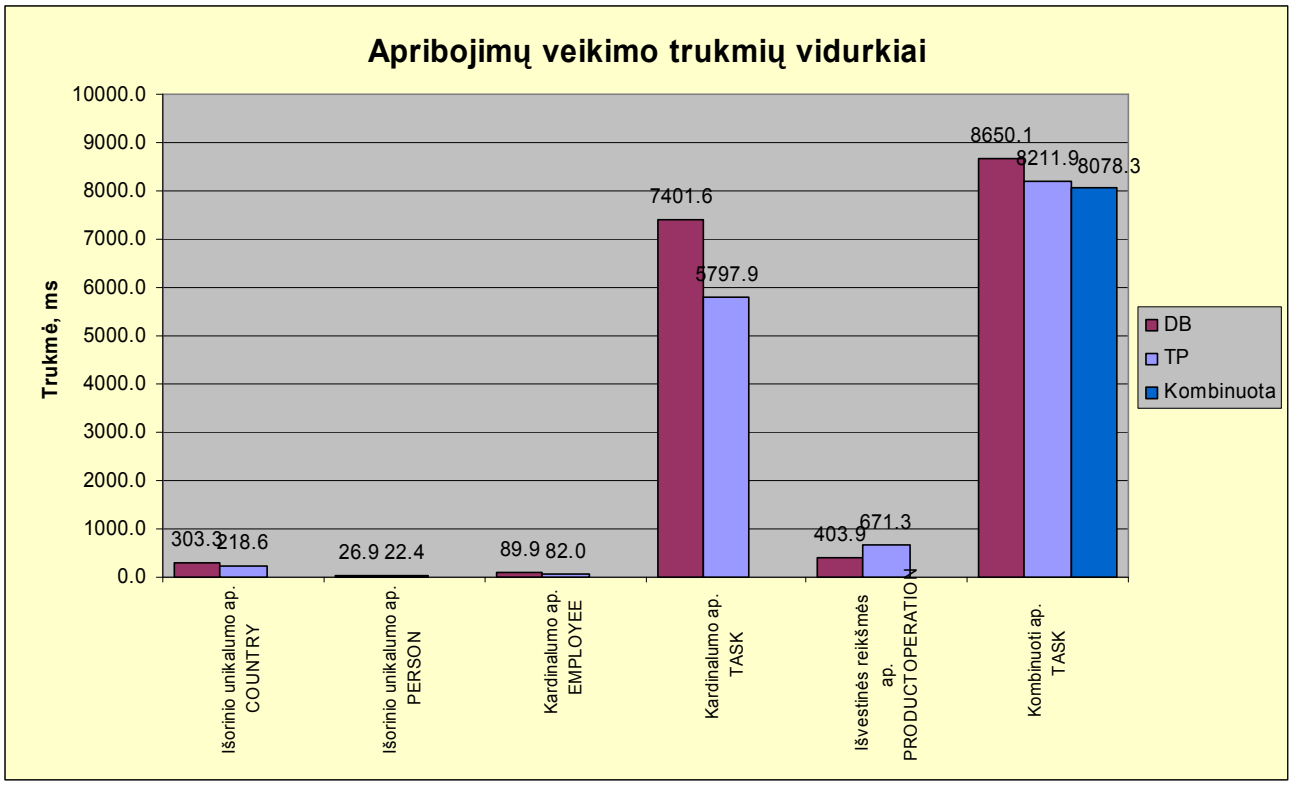
Dar viena svarbi užklausos dalis – lentelių sujungimai JOIN komanda. Jungiant lenteles INNER jungtimi, užklausa kur kas greičiau veikia, jeigu visas prijungiamos lentelės laukų reikšmių sąlygas tikrinsime prijungimo, o ne WHERE sakinyje. Taip yra dėl to, kad šiuo atveju prie pirminės lentelės bus prijungiama tik tiek įrašų, kiek tenkina nurodytą sąlygą. Priešingu atveju prie lentelės būtų jungiami visi laukų reikšmėmis sutampantys įrašai ir tik tada būtų atmetami įrašai, netenkinantys sąlygos.

Dažnai paieškos rezultatus norima gauti tvarkingus, t.y. surūšiuotus, bet šis procesas irgi nemažai atima laiko, kadangi rūšiuojant lyginamos visos gražintinos reikšmės tarpusavyje ir dėliojama eilės tvarka. Duomenų rūšiavimas grąžinant 1000 įrašų prailgina užklausos laiką 6 kartus, o grąžinant 10 000 įrašų – 13 kartų.

Nemažą įtaką duomenų radimui rentelėje turi lentelių laukų indeksavimas. Indeksas yra surūšiuota aibė nuorodų į lentelės eilutes. Tai atskiras duomenų bazės objektas ir jis nėra saugomas lentelėje. Kai sukuriamas indeksas DBVS automatiškai sukuria šią struktūrą ir palaiko ją, t.y. seka, kad indeksas, besikeičiant bazinės lentelės duomenims, nuolat atspindėtų jos duomenis ir kad indeksas būtų surūšiuotas. Jeigu turime sukūrę indeksą laukui, DBVS gali panaudoti jį įrašo eilutės su konkrečia lauko reikšme vietai nustatyti, vietoj to kad paeiliui tikrinti kiekvieną lentelės įrašą. Palyginimui, reikšmės paieškos trukmė indeksuotame lauke lentelėje su 600 000 įrašų užtruko tiek pat, kiek ir neindeksuotoje lentelėje su 6 įrašais, t.y. kelias milisekundes. Toje pačioje lentelėje su 600 000 įrašų reikšmės paieška neindeksuotame lauke užtruko virš pusės minutės.

4.6. Eksperimentinio tyrimo apibendrinimas

Eksperimentui buvo pasirinkti 3 skirtingi apribojimų tipai, kadangi jų veikimo charakteristikos labiausiai priklauso nuo realizacijos būdo. Jais realizuoti ir ištirti 5 atskiri bei vienas kombinuotas apribojimas. Likusių apribojimų veikimas trunka itin trumpai, dėl to sunku išmatuoti tikslią operacijų trukmę. Eksperimento metu gauti matavimų rezultatai pateikti 20 paveikslėlyje.



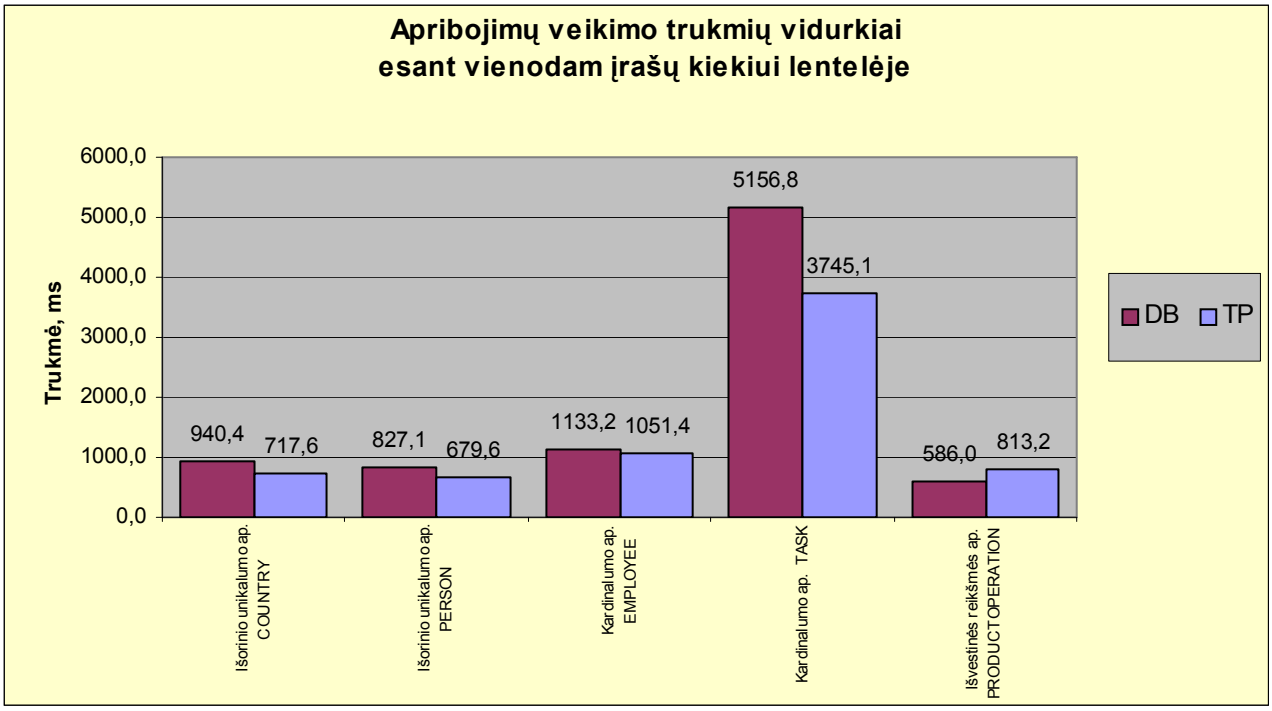
20 pav. Apribojimų veikimo trukmių vidurkiai

Kaip matome iš rezultatų, išorinio unikalumo ir kardinalumo apribojimų veikimo trukmės vidurkis realizuojant jį trigeryje yra 4-100 ms didesnis, nei tikrinimą atliekant prieš įterpinėjant įrašą. Jeigu operacijos metu vykdomos 3-5 užklausos, o operuojama lentelėmis, kuriose yra milijonai įrašų, šis skirtumas gali padidėti ir iki kelių sekundžių. Tai palyginus nedidelis skirtumas, tačiau, jeigu sistemoje vienos operacijos metu bus vykdomas ne vienas apribojimas, o jų kombinacija, galima sutaupyti iki 10 sekundžių.

Jeigu apribojimas reikalingas išvestinei reikšmei apskaičiuoti, kuri bus terpiama į kitą lentelę, apribojimas trigeryje suveikia netgi greičiau, negu jis veiktų taikomojoje programoje, kadangi pastaruoju atveju mano eksperimentinėje sistemoje į duomenų bazę reikia siųsti mažiausiai tris užklausas: pirma – duomenų, reikalingų skaičiavimams atlikti ištraukimas, antra – įrašo įterpimas/atnaujinimas į pirminę lentelę, trečia – apskaičiuoto įrašo įterpimas/atnaujinimas į antrinę lentelę. Šiuo atveju apribojimas trigeryje, kur visos operacijos atliekamos jo viduje, veikia 200-300 ms trumpiau.

Situacija gali pasikeisti, jeigu išvestinei reikšmei apskaičiuoti reikės sudėtingo algoritmo ir funkcijų, tuomet galima daryti prielaidą, kad apribojimas taikomojoje programoje veiks sparčiau, kadangi visi skaičiavimai bus atliekami kiekvieno vartotojo kompiuterio resursų sąnaudomis. Eksperimentas tokiam atvejui nebuvo vykdomas, kadangi šiai eksperimentinei sistemai nepavyko pritaikyti sudėtingų skaičiavimų, kurių trukmės būtų akivaizdžiai didesnės.

Norint palyginti visų šių apribojimų veikimo trukmes tarpusavyje, reikia kiekvieno apribojimo tyrimui sudaryti panašias sąlygas. Įrašų kiekiai lentelėse varijuoja labai skirtingai, nuo 5000 iki 10 000 000, dėl to eksperimento metu gauti rezultatai yra labai įvairūs ir nesulyginami. Tokiu tikslu visose šiose lentelėse įrašų kiekiai suvienodinami iki 1 000 000 įrašų. Tokiomis sąlygomis tiriant kiekvieną apribojimą, realizuojant jį tiek trigeriu, tiek ir apribojimų valdikliu, imami imčių iš 5 elementų matavimų trukmių aritmetiniai vidurkiai. Jų visų palyginimas matomas 21 paveikslėlyje.



21 pav. Apribojimų veikimo trukmių vidurkiai esant vienodam įrašų kiekiui lentelėje

Kaip matyti iš grafiko, visų apribojimų tipų veikimo trukmės svyruoja tarpusavyje nedideliu skirtumu, tačiau kardinalumo apribojimas TASK lentelei vis dar ilgai vykdomas. Tai yra dėl to, kad įterpinėjant įrašą į šią lentelę, kreipiamasi į dar tris skirtingas lenteles, dėl to vykdymo laikas gerokai viršija kitų duomenų vientisumo apribojimų tipu veikimo trukmes.

5. Duomenų vientisumo apribojimų realizacijos strategijos parinkimo metodikos apibendrinimas

Analizė rodo, kad organizacijos gali pasirinkti įvairius vientisumo apribojimų realizavimo variantus:

- Visus apribojimus realizuoti duomenų bazėje;
- Visus apribojimus realizuoti taikomojoje programoje;
- Realizuoti apribojimus ten, kur jų veikimas efektyvesnis – dalį duomenų bazėje, dalį – taikomojoje programoje.

Pirmu atveju toks realizavimo būdas galėtų būti įgyvendinamas įmonėse, kurios naudojasi viena pastovia duomenų baze. Tuomet visi apribojimai būtų struktūrizuoti vienoje vietoje, neišmėtyti sistemoje, taip supaprastinant jų kūrimą ir priežiūrą. Taip pat, esant reikalui, nesudėtinga būtų visus duomenų apribojimus perkelti į kitą duomenų bazę, nesuardant jų struktūrų bei neprarandant atskirų jų dalių.

Antru realizavimo būdu vertėtų kurti sistemas, kurios turi ne vieną duomenų bazę, arba jos dažnai yra keičiamos, pereinant nuo vienos prie kitos. Tuomet visi duomenų apribojimai būtų sukoncentruoti viename apribojimų valdiklyje. Kitas pranašumas yra tas, kad duomenų bazių apribojimai yra palyginti ribotos adaptacijos, t.y. ne visada gali užtekti priemonių, norint sukurti duomenų vientisumo apribojimą išskirtiniam nestandartiniam atvejui. Šiuo atžvilgiu apribojimų valdiklis yra kur kas lankstesnis ir labiau pritaikomas išskirtiniams poreikiams.

Paskutiniojo vientisumo apribojimų realizavimo varianto strategijos parinkimas buvo atliekamas eksperimentu tiriant bei nustatant duomenų vientisumo apribojimų tipų veikimo charakteristikas, esnat skirtingiems realizavimo būdams. Iš eksperimento metu gautų rezultatų galima išskirti šiuos duomenų vientisumo apribojimų realizacijos strategijos parinkimo principus, projektuojant, kuriant ir palaikant dideles duomenų bazes:

- Pirmiausia reikia sudaryti duomenų bazės schemą su vientisumo apribojimais.
- Pirminio, išorinio rakto, vidinio unikalumo apribojimus realizuoti DB.
- Išorinio unikalumo ir kardinalumo apribojimus realizuoti valdikliu.
- Išvestinės reikšmės arba didelias programines operacijas atliekantiems apribojimams atlikti eksperimentą.
- Jei yra daug apribojimų ir įvairių jų kombinacijų, tikslinga eksperimento būdu nustatyti, kokiomis priemonėmis ir konkrečiomis realizacijomis įgyvendinti tokius apribojimus.

6. Išvados

- Darbo su didelėmis DB praktika rodo, kad vientisumo apribojimų užtikrinimo efektyvumas yra problema. Kuris apribojimų realizavimo būdas yra pranašesnis, vienareikšmiškai negalima atsakyti, tam reikia rasti tinkamą realizacijos būdų kombinaciją.
- Atlikta skirtingų apribojimų tipų realizavimo galimybių analizė leido daryti preliminarias hipotezes, kaip geriau juos realizuoti – taikomojoje programoje ar duomenų bazėje.
- Atliktas eksperimentas patvirtino, kad preliminarios hipotezės buvo teisingos, ir leido nustatyti statistines apribojimų užtikrinimo priemonių veikimo charakteristikas.
- Remiantis eksperimentiniu tyrimu, buvo sudarytos apribojimų realizavimo strategijos pasirinkimo rekomendacijos ir metodika, kaip realizuoti tam tikrų tipų apribojimus ir atlikti eksperimentą tinkamiausiai strategijai nustatyti.
- Atlikti tyrimai leidžia daryti išvadą, kad tinkamiausio duomenų vientisumo apribojimų realizacijos būdo pasirinkimas turi būti taikomas kartu su užklausų optimizavimu tam, kad gauti geriausią galimą veikimą.

7. Literatūra

1. Morgan Kaufmann, SQL Practical Guide For Developers, 2005
2. Baronas R., Duomenų Bazių Valdymo Sistemos, 2005
3. Mamajev J., Microsoft SQL Server 2000: V Podlinnike, 2005
4. Connolly T.M, Begg C.E., Database Systems. A Practical Approach To Design, Implementation, And Management. Addison-Wesley, 3th ed., 2003.
5. Date C.J., Introduction To Database Systems. Reading, Mass.: Addison-Wesley, 7th ed., 2003.
6. Cochrane R, Pirahesh H, Mattos N. Integrating Triggers and Declarative Constraints in SQL Database Systems. VLDB. Mumbai (Bombay), India, 1996
7. ISO/IEC 9075:1999. Information Technology – Database Languages – SQL – part 2: Fondation, 1999
8. Scott Noyes.Triggers in MySQL, Oracle and SQL Server. 2006, gruodis [žiūrėta 2007-01-10]. Prieiga per Internetą: http://searchopensource.techtarget.com/loginMembersOnly/1,289498,sid39_gci1231331,00.html
9. Jim McBeath. Oracle and SQL Server triggers. 2002, vasaris [žiūrėta 2007-01-10]. Prieiga per Internetą: http://www.mckoi.com/database/maillist/msg00412.html
10. Wikipedia. Database trigger. [žiūrėta 2007-01-10]. Prieiga per Internetą: http://en.wikipedia.org/wiki/Database_trigger
11. Alex Bunardzic. The Myth of Data Integrity. 2005, lapkritis [žiūrėta 2007-01-15]. Prieiga per internetą: http://jooto.com/blog/?p=22
12. John Charles Olamendy. Query Optimization in SQL Server 2005. [žiūrėta 2007-05-14]. Prieiga per Internetą: http://www.c-sharpcorner.com/UploadFile/john_charles/QueryoptimizationinSQLServer20051211200715430 3PM/QueryoptimizationinSQLServer2005.aspx
13. Elmasri, Navathe**.** Fundamentals of Database Systems*. 3 chapter.*
14. Oracle Developer Team. Data Integrity. [žiūrėta 2007-05-14]. Prieiga per Internetą: http://support.cs.nott.ac.uk/help/docs/databases/oracle/standard/server.101/b10743/data_int.htm
15. Wikipedia. Data integrity. [žiūrėta 2007-05-14]. Prieiga per Internetą: http://en.wikipedia.org/wiki/Data_integrity
16. David C. Hay. A comparison of data modeling techniques. [žiūrėta 2007-11-23] Prieiga per Internetą: http://www.essentialstrategies.com/publications/modeling/compare.htm
17. Oracle Corporation. Maintaining Data Integrity Through Constraints. [žiūrėta 2007-11-23] Prieiga per Internetą: http://download.oracle.com/docs/cd/B10500_01/appdev.920/a96590/adg05itg.htm