KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Edvinas Eidukevičius

VIENTISUMO APRIBOJIMŲ ĮGYVENDINIMO DUOMENŲ BAZĖSE METODIKA

Magistro darbas

Vadovas
doc. dr. Lina Nemuraitė

KAUNAS, 2006

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

VIENTISUMO APRIBOJIMŲ ĮGYVENDINIMO DUOMENŲ BAZĖSE METODIKA

Magistro darbas

Vadovas
doc. dr. L. Nemuraitė

Recenzentas
dr. D. Makackas

Atliko
IFM 0/4 gr. stud.
E. Eidukevičius

KAUNAS, 2006

Turinys

Santrauka

VIENTISUMO APRIBOJIMŲ ĮGYVENDINIMO DUOMENŲ BAZĖSE METODIKA

Darbe svarstomos sudėtingų taisyklių, vientisumo apribojimų realizavimo galimybės. Pateikti išskirti vientisumo apribojimai ir apribojimų tipai, pateiktas jų vaizdavimas UML, bei jų kodas ORACLE PL/SQL sistemoje. Išanalizuotos apribojimų realizavimo galimybės. Taip pat realizuoti ir ištestuoti visų tipų apribojimai duomenų bazėje. Pateikta metodinė medžiaga padėsianti vartotojui įsisavinti apribojimų veikimo principus jų realizavimo galimybes. Sukurta sistema pateikianti informaciją apie apribojimus, jų šablonus. Kai kuriems apribojimams sistema pati generuoja PL/SQL kodą vartotojui pasirinkus atitinkamus parametrus.

Summary

Realization of methodology of integrity constrains in dabatabases

This work discuses complex rule systems, integrity constraints implementation issues. It describes all integrity constraints and their types. Showed their representation in UML concepts also ORACLE PL/SQL system code. Analysed realization possibilities of integrity constraints. Realized in database system and enforced testing process in this database of integrity constraits. Introduces users helpfull methodology about peration issues, possibilities and methods. Managed system of integrity constains creating templates. In some ways program generates PL/SQL code by users defined variables.

1 Įvadas

Duomenų bazių sistemų atsiradimas sudarė dideliam naudotojų kiekiui galimybę vienu metu efektyviai ir kontroliuojamu būdu naudotis dideliais duomenų kiekiais. Įprastos duomenų bazių valdymo sistemos (DBVS) buvo pasyvios. Jose duomenys buvo kuriami, pateikiami naudotojams, modifikuojami arba naikinami naudotojų arba dalykinių programų komandomis, kurios nustatytos SQL2 standarte. Pažangios DBVS turi išplėstą funkcionalumą ir tapo aktyvios, t.y. DBVS tam tikrus veiksmus vykdo automatiškai reaguodama į tam tikrus įvykius arba susidariusias sąlygas. SQL3 standarte numatyti trigeriai arba dar kitaip vadinamos aktyvios taisyklės, kurias galima traktuoti kaip produkcinių taisyklių realizaciją DBVS.

Trigeriai yra aktyvus DBVS elementai, kurie aktyvuojasi susidarius tam tikrom sąlygoms ir užtikrina platų funkcionalumo spektrą, kuris apjungia ir darnos suvaržymų realizaciją, išvestinių duomenų tvarkymą, duomenų saugumą. duomenų evoliuciją ir priežiūrą ir pan.

ADBVS yra iš esmės susijusios su taisyklių sąvoka. Pačios taisyklės yra nustatomos naudotojų, dalykinių programų, ir/arba DB administratorius. Visi jie nustato reikalaujamą aktyvią elgseną. Pačiu bendriausiu požiūriu aktyvios duomenų bazės taisyklės sudarytos iš trijų daliu: *Įvykis, Sąlyga* ir *Veiksmas*. Angliškai tokio tipo taisyklės sutrumpintai vadinamos ECA nuo *Event, Condition, Action*. *Įvykis* sąlygoja taisyklės veikimo pradžią. *Sąlyga* tikrinama taisyklės veikimo pradžioje ir *Veiksmas* yra vykdomas kai įvyksta *Įvykis* ir *Sąlyga* yra teisinga. ADBVS yra DBVS, kurioje įgyvendintas ECA modelis ir užtikrintas ECA taisyklių modelio vykdymas. Viena iš priežasčių, kodėl ADBVS nėra plačiai naudojamos praktikoje yra nepakankamas trigerių taikomųjų programų kūrimo palaikymas. Akivaizdžiai trūksta priemonių, remiančių tokių programų kūrimą, kurio metu būtina užtikrinti taisyklių vertinimą ir patikrą kartu neišleidžiant iš akių tipinių taisyklėms, jų konfliktų ir tarpusavio priklausomybių, problemų.

Programuotojų, projektuotojų, DB administratorių problemos kuriant sudarinėjant duomenų bazės apribojimus bei taisykles. Kadangi akivaizdžiai trūksta priemonių, kurios leistų užtikrinti apribojimų realizavimą, peržiūrą ir patikrinimą. Vartotojams sunkiai įsivaizduojama apribojimų taisyklių vykdymo semantika, kadangi nėra tiksliai specifikuota, kaip vienų ar kitų atveju užtikrinti duomenų apribojimus.

Analizės tikslai buvo išsiaiškinti sistemose vyraujančius vientisumo apribojimus, įvertinti apribojimų realizavimo galimybes duomenų bazių valdymo sistemose, apžvelgti apribojimų užtikrinimo realizavimo ir automatizavimo galimybes.

Pirminio identifikatoriaus, būtinumo, unikalumo apribojimai yra svarbiausi, kurie yra naudojami visuose modeliavimo metoduose ir diegiami DBVS. Galimybė naudoti šiuos apribojimus priklauso nuo to, kokios ryšio tipus naudojame konceptualiame modelyje. Tiktai kardinalumo ir nuorodos apribojimai yra bendri visuose konceptualiuose modeliuose. Apžvelgiant aukščiau pateiktą lentelę, galime padaryti išvadas, kad ORM modelis yra galingiausias ir išraiškingiausias modelis, skirtas apribojimų modeliavimui. Tačiau iš kitos pusės, šis modelis yra pakankamai naujas ir nepasižymi dideliu CASE įrankių asortimentu, kurie leistu palaikyti ši modelį ir galėtų perkelti į kitus modelius, tokius kaip UML, XML.

Lyginant su ORM, UML yra palaikomas daugelio CASE įrankių ir priimtas kaip modeliavimo kalbos standartas. Kadangi UML yra labai išplečiantis, tai pasistengsime išplėsti UML modelį su stereotipais skirtais apribojimams, kurie bus apžvelgiami visuose kituose modeliuose. Sekanti naudinga UML savybė yra galimybė apibūdinti apribojimus objekto apribojimų kalboje kaip pastebėjimą apribojančiam objektui ir

kituose etapuose perkelti juo automatiškai į konkrečios DBVS duomenų struktūras kaip CHECK funkcijas, trigerius ir saugomas procedūras.

Nėra vieningų taisyklių, kaip realizuoti įvairių tipų apribojimus. DBVS naudojimo metodikos pateikia taisykles tik pagrindinių apribojimų realizavimui. Todėl projektuotojai ir programuotojai apsiriboja paprastais apribojimais, o kitus realizuoja programinėmis priemonėmis. Neišnaudojamos DBVS galimybės, kurios leistų pagerinti programų korektiškumą ir laiko charakteristikas, kadangi apribojimų realizavimas taikomosiose programose iššaukia galimus nesuderinamumus ir ilgesnį vykdymo laiką. Nėra DBVS realizuotų apribojimų peržiūros ir tikrinimo priemonių. Šiame darbe bus sprendžiama tik dalis problemų, susijusių su apribojimų įgyvendinimu DBVS. Tai yra, bus siekiama rasti geriausius įvairių tipų apribojimų realizavimo būdus ir pateikti juos vartotojams kaip pakartotinio naudojimo šablonus.

Paprasčiausia ir dažniausiai naudojama apribojimų (constraints) realizavimo idėja yra naudoti išorinius raktus ir CHECK funkcijas, tačiau šių galimybių nepakanka. Apribojimų realizavimui pasirinkta ORACLE 10g duomenų bazių valdymo sistema pilnai atitinkanti daugelį reikalavimų. ORACLE 10g DBVS kai kuriuos paprastus apribojimus palaiko automatizuotai: pirminis atributo identifikatorius, atributo identifikatorius, nurodomasis atributo apribojimas, būtinasis atributo apribojimas, vidinis unikalumo apribojimas. Kitų likusių apribojimų realizavimui panaudosime DBVS palaikančiomis CHECK funkcijomis, TRIGERIAIS ir PAKETAIS.

Iš viso galima išskirti keliolika apribojimų tipų, vieni realizuojami kuriant trigerius, check funkcijas, kiti yra tiesiog automatizuoti pačiose duomenų bazių valdymo sistemose. Pateiksime apribojimų realizavimo galimus sprendimo būdus.

Darbo eigoje buvo aprašyti visi vientisumo apribojimai, pateiktas jų vaizdavimas UML, bei jų kodas ORACLE sistemoje, realizuoti visų tipų apribojimai duomenų bazėje, pateikta metodinė medžiaga padėsianti vartotojui įsisavinti apribojimų veikimo principus jų realizavimo galimybes, sukurta sistema pateikianti informaciją apie apribojimus, jų šablonus. Kai kuriems apribojimams sistema pati generuoja PL/SQL kodą vartotojui pasirinkus atitinkamus parametrus.

Darbo tikslas pateikti geriausius apribojimų būdus kaip šablonus, sukurti sistemą šablonams peržiūrėti ir ten, kur įmanoma, SQL kodui generuoti, aprašyti testavimo duomenų pavyzdžius.

Šis darbas padės metodikos vartotojams, projektuotojams, duomenų bazių administratoriams susipažinti su apribojimų specifikacijomis, panaudojimu bei realizavimu, kadangi iki šiol nebuvo jokių palengvinančių priemonių šioje srityje.

2 Vientisumo apribojimų ir jų užtikrinimo galimybių analizė

2.1. Analizės tikslas

Analizės tikslai:

- Išsiaiškinti sistemose vyraujančius vientisumo apribojimus
- įvertinti apribojimų realizavimo galimybes duomenų bazių valdymo sistemose
- apžvelgti apribojimų užtikrinimo automatizavimo galimybes.

2.2 Tyrimo sritis, objektas ir problema

Tyrimo sritis – vientisumo apribojimai ir jų naudojimo galimybės DBVS.

Objektas – vientisumo apribojimų realizavimo DBVS metodika.

Problemos:

- Nėra vieningų taisyklių, kaip realizuoti įvairių tipų apribojimus. DBVS naudojimo metodikos pateikia taisykles tik pagrindinių apribojimų realizavimui. Todėl projektuotojai ir programuotojai apsiriboja paprastais apribojimais, o kitus realizuoja programinėmis priemonėmis.
- Neišnaudojamos DBVS galimybės, kurios leistų pagerinti programų korektiškumą ir laiko charakteristikas, kadangi apribojimų realizavimas taikomosiose programose iššaukia galimus nesuderinamumus ir ilgesnį vykdymo laiką.
- Nėra DBVS realizuotų apribojimų peržiūros ir tikrinimo priemonių.

Šiame darbe bus sprendžiama tik dalis problemų, susijusių su apribojimų įgyvendinimu DBVS. Tai yra, bus siekiama rasti geriausius įvairių tipų apribojimų realizavimo būdus ir pateikti juos vartotojams kaip pakartotinio naudojimo šablonus.

3.1 Vartotojų aibė, tipai ir savybės

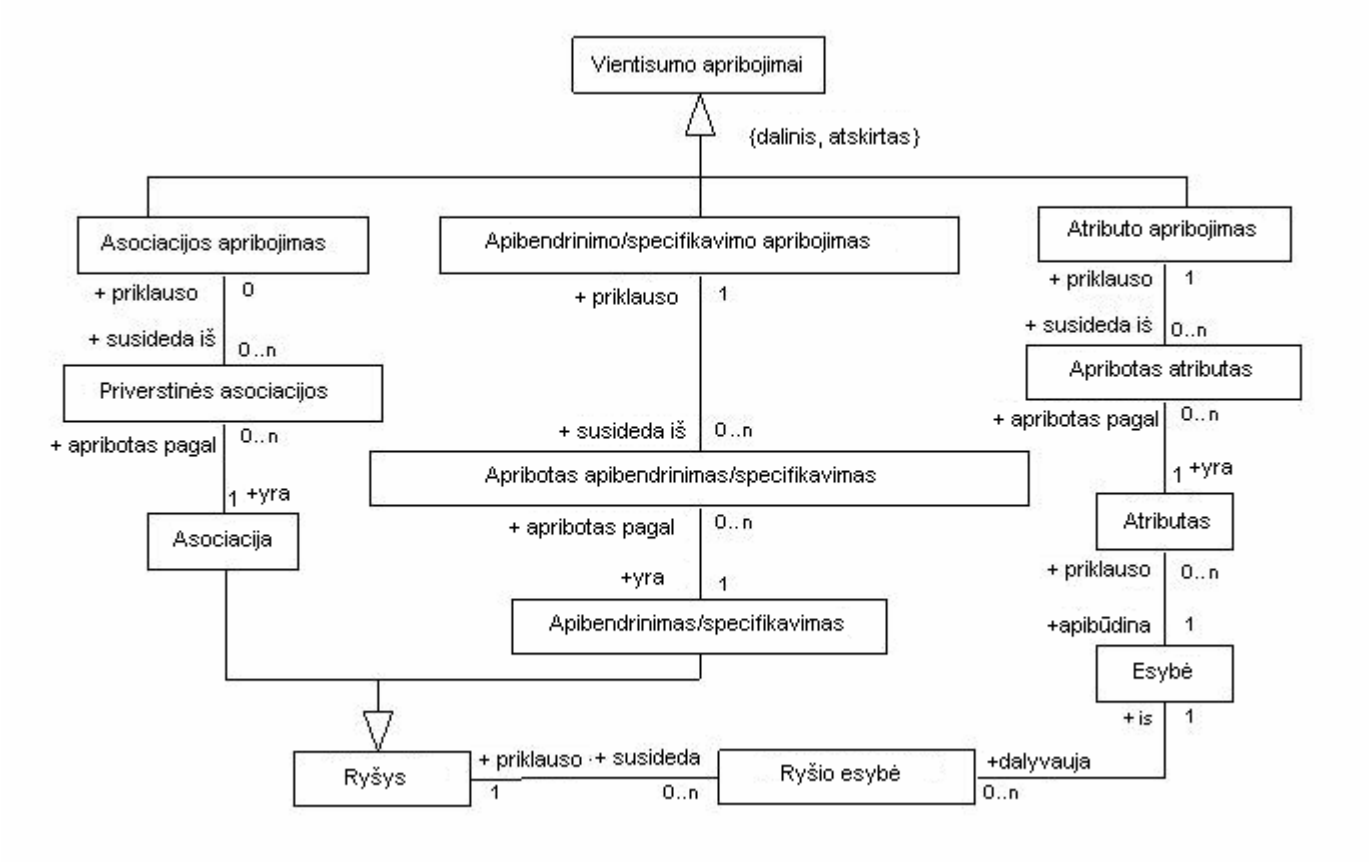
Metodikos vartotojai – projektuotojai, DB administratoriai, programuotojai.

3.2 Vartotojų tikslai ir problemos

Programuotojų, projektuotojų, DB administratorių problemos kuriant sudarinėjant duomenų bazės apribojimus bei taisykles. Kadangi akivaizdžiai trūksta priemonių, kurios leistų užtikrinti apribojimų realizavimą, peržiūrą ir patikrinimą. Vartotojams sunkiai įsivaizduojama apribojimų taisyklių vykdymo semantika, kadangi nėra tiksliai specifikuota, kaip vienų ar kitų atveju užtikrinti duomenų apribojimus.

3.3. Vientisumo apribojimų nagrinėjimas konceptualaus modeliavimo metoduose

Konceptualus modeliavimas yra svarbus žingsnis kuriant informacinę sistemą. Praktikoje naudojami grafiniai modeliai negali išreikšti visų dalykinės srities savybių, todėl jie papildomi tekstinėmis išraiškomis – vientisumo apribojimais. Pagrindinės apribojimų sąvokos parodytos 1 paveiksle:



1 pav. Pagrindiniai apribojimų konceptai ir ryšiai tarp jų

Vientisumo apribojimai išskirstyti į asociacijos, apibendrinimo/specifikavimo bei atributo apribojimus. Apribojimas susideda iš priverstinų reikalavimu ir taisyklių. Pavyzdžiui, turima esybė, kuriai priklauso apibūdinti atributai. Apibūdinti atributai gali turėti įvairiausių priverstinų taisyklių ar apribojimų,

visa tai sueina į atributo apribojimo konceptą ir į patį aukščiausią lygį - vientisumo apribojimų. Kaip ir matyti 1 paveiksle.

Įvairūs konceptualaus modeliavimo metodai leidžia aprašyti skirtingus apribojimų tipus. 3 paveiksle pateikta apribojimų tipų klasifikacija, kuri apima visą apribojimų įvairovę. Šie apribojimų tipai parodyti 2 paveikslo lentelėje. Palyginimui parinkti 7 modeliavimo metoduose naudojami apribojimai. ER, EER, HERM, ORM, UML, xUML, RDBMS.

Apribojimo tipas\ Konceptualaus modeliavimo metodas	ER	EER	HERM	ORM	UML	xUML	RDBMS
Apribojimas vienam atributui							
Reišmių/rinkinio (Value/set of values)	–	–	√	√	√	√	√
Privalomas/nebūtinas (Mandatory/optional)	–	√	√	√	√	√	√
Įgytas (Derived)	–	–	–	–	√	√	–
Apribojimai atributų atributams/grupei							
Pirminio identifikatoriaus (Primary identifier)	√	√	√	√	–	√	√
Identifikatoriaus (Identifier)	–	√	–	–	–	√	–
Unikalumo (Uniqueness)	–	–	–	√	–	√	√
Išorinio unikalumo (External uniqueness)	–	–	–	√	–	–	√
Nurodomojo vientisumo (Referential integrity)	√	√	√	√	√	√	√
Apribojimai nebūtinų atributų grupei							
Lygių rinkinių (Equal sets)	–	√	–	√	–	–	–
Išskyrimo (Exclusion)	–	√	–	√	–	–	–
Poaibio (Subset)	–	–	√	√	–	–	–
Skiriamojo būtinumo (Disjunctive mandatory)	–	√	–	√	–	–	–
Apribojimai ryšiams							
Daugialypikumo (Multiplicity)	√	√	√	√	√	√	√
Priešingybės/priklausymo (Inverse/inclusion)	–	√	–	–	–	–	–
Įgytas (Derived)	–	–	–	–	√	√	–
Apribojimai ryšių grupėms ir keliams (vėliau ORM pavadintos kaip sujungimo apribojimai)							
Lygių rinkinių (Equal sets)	–	–	–	√	–	√	–
Išskyrimo (Exclusion)	–	–	√	√	–	–	–
Poaibio (Subset)	–	–	√	√	–	√	–
Skiriamojo būtinumo (Disjunctive mandatory)	–	–	–	√	–	–	–
Aciklišku kilpai (Acyclic loop)	–	–	–	–	–	√	–
Apribojimai refreksyviems ryšiams							
Irefleksivus (Ireflexive)	–	–	–	√	–	–	–
Asimetrinis (Asymmetric)	–	–	–	√	–	–	–
Aciklinis (Acyclic)	–	–	–	√	–	√	–
Intranzityvinis (Intransitive)	–	–	–	√	–	–	–
Simetrinis (Symmetric)	–	–	–	√	–	–	–
Asimetrinis (Antisymmetric)	–	–	–	√	–	–	–
Apribojimai ryšių apibūdinimo/specifikavimo							
Išskyrimo (Disjoint)	–	√	√	√	√/–	√	–
Pilnumo (Complete)	–	√	√	√	√/–	√	–

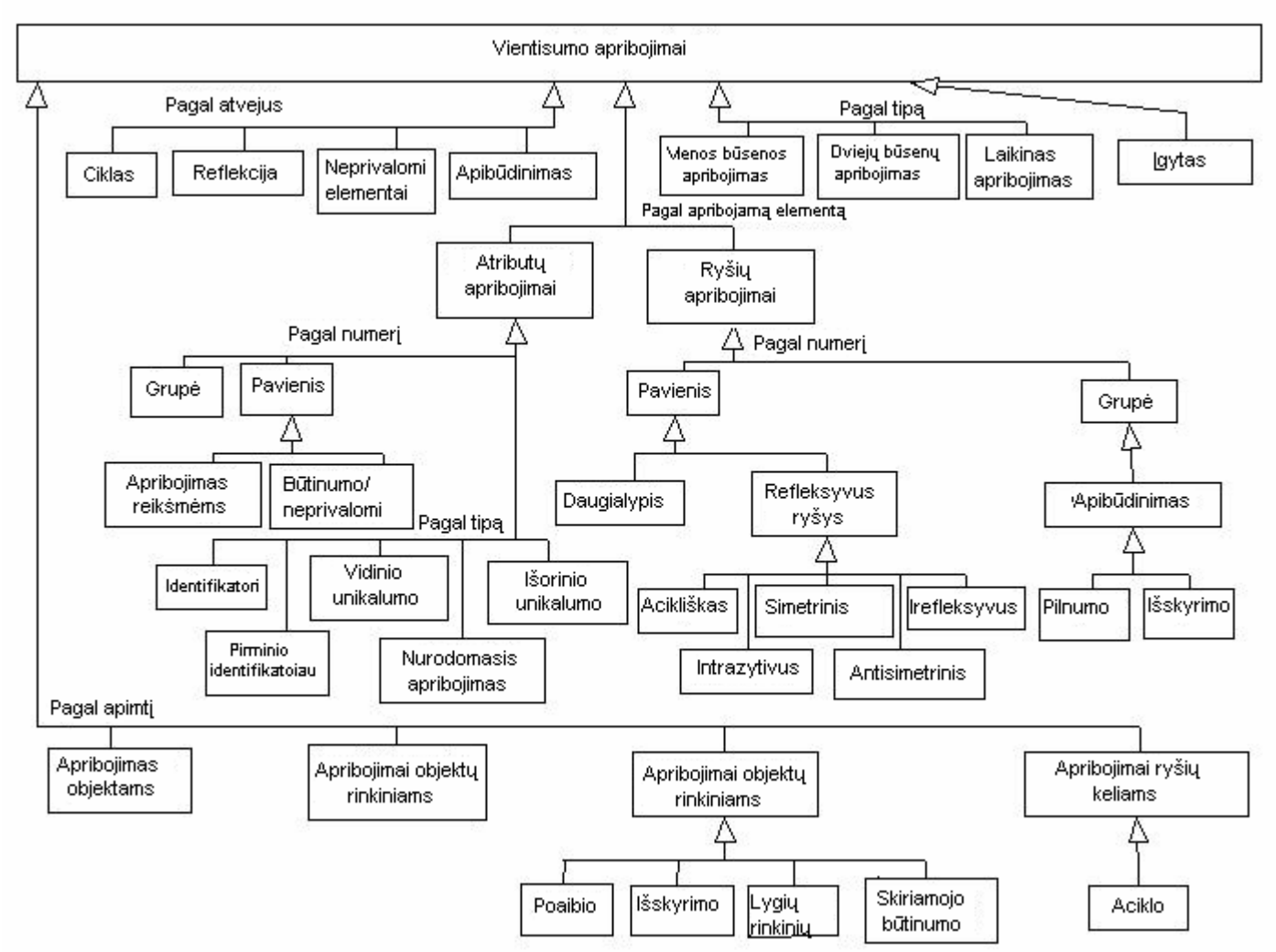
2 pav. Vientisumo apribojimai naudojami skirtinguose konceptualaus modeliavimo metoduose

Pirminio identifikatoriaus, būtinumo, unikalumo apribojimai yra svarbiausi, kurie yra naudojami visuose modeliavimo metoduose ir diegiami DBVS. Galimybė naudoti šiuos apribojimus priklauso nuo to, kokios ryšio tipus naudojame konceptualiame modelyje. Tiktai kardinalumo ir nuorodos apribojimai yra bendri visuose konceptualiuose modeliuose. Apžvelgiant aukščiau pateiktą lentelę, galime padaryti išvadas, kad ORM modelis yra galingiausias ir išraiškingiausias modelis, skirtas apribojimų modeliavimui. Tačiau iš kitos pusės, šis modelis yra pakankamai naujas ir nepasižymi dideliu CASE įrankių asortimentu, kurie leistu palaikyti ši modelį ir galėtų perkelti į kitus modelius, tokius kaip UML, XML.

Lyginant su ORM, UML yra palaikomas daugelio CASE įrankių ir priimtas kaip modeliavimo kalbos standartas. Kadangi UML yra labai išplečiantis, tai pasistengsime išplėsti UML modelį su stereotipais skirtais apribojimams, kurie bus apžvelgiami visuose kituose modeliuose. Sekanti naudinga UML savybė yra galimybė apibūdinti apribojimus objekto apribojimų kalboje kaip pastebėjimą apribojančiam objektui ir kituose etapuose perkelti juo automatiškai į konkrečios DBVS duomenų struktūras kaip CHECK funkcijas, trigerius ir saugomas procedūras.

3.4. Vientisumo apribojimų klasifikacija

Vientisumo apribojimai yra suskirstyti i kategorijas, atsižvelgiant į keletą kriterijų:

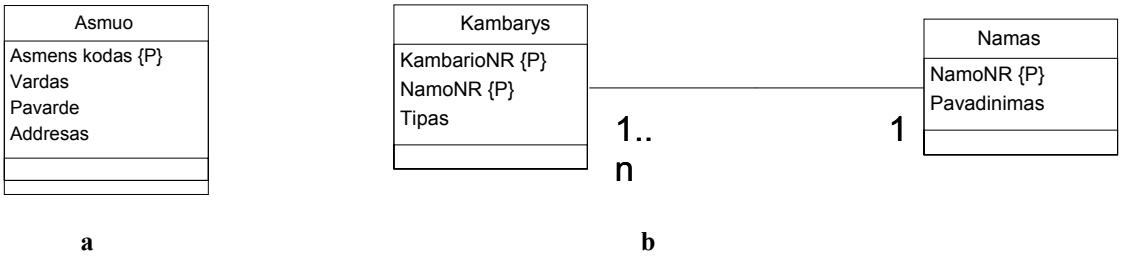


3 pav. Apribojimų kategorijos

- **Pirminis identifikatorius**

Šis apribojimas naudojamas unikaliam atpažinimui. Yra reikalaujama, kad identifikuojamas atributas ar atributų grupė visada turėtų reikšmes ir tos reikšmės turi būti unikalios.

Apribojimo pavyzdys. Esybės „Asmuo“ identifikatorius gali būti atributas „Asmens kodas“, kadangi jis unikaliai identifikuoja asmenį. Esybės „Kambarys“ identifikatorius „KambarioNr“ negali būti, kadangi jis yra unikalus per visus kambario numerius ir to pačio namo. Identifikatorius esybės „Kambarys“ gali būti atributai „KambarioNr“ ir „NamoNr“.



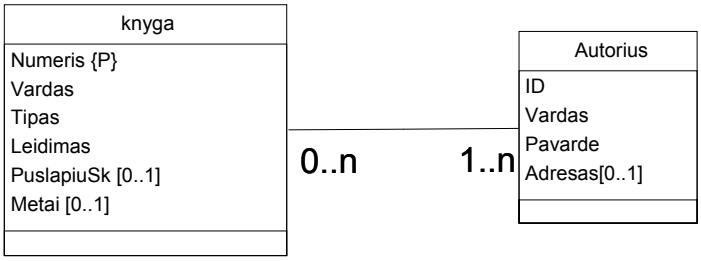
Pirminis identifikavimas apribojimams atributui (a) ir atributu grupei (b).

4 pav. Pirminio identifikatoriaus apribojimas

- **Paprasto būtinumo apribojimas**

Šis apribojimas yra naudojamas atributui ar esybės tipo objekto vaidmeniui nustatyti koks atributas turi turėti reikšmę arba kuri vaidmuo turi būti atliekamas pagal visus užpildymo reikalavimus

Naudojimas. Apribojimas yra naudojamas atributui ar ryšiui tarp esybių.
Apribojimo pavyzdys. Esybės tipo objektas „Knyga“ yra identifikuojama pagal knygos atributą „Numeris“ ir turi tris paprasto būtinumo atributų apribojimus. Tai reiškia, kad visi esybės „Knyga“ reikalavimai turi turėti reikšmes atributams „Vardas“, „Tipas“ ir „Leidimas“.
Pažymėti, kad kiekviena knyga, turi turėti autorių, mes turime naudoti būtinumo vaidmens apribojimą. Tai reiškia, kad esybės „Knyga“ pavyzdžiai turi turėti ryšį su esybės „Autorius“ pavyzdžiais ir negali egzistuoti esybės „Knyga“ be susiejamo ryšio su esybės „Autorius“.



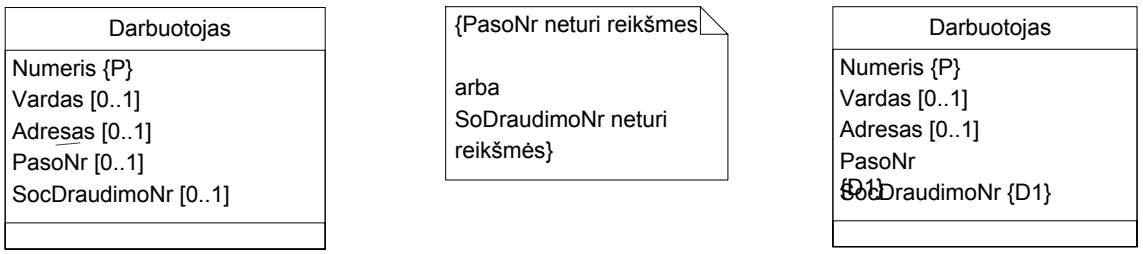
Paprasto būtinumo apribojimai attributams ir susiejimui.

5 pav. Paprasto būtinumo apribojimas

- **Skiriamojo būtinumo apribojimas**

Šis apribojimas reiškia, kad perskyrimas esybės atributų yra būtinas. Tai reiškia, kad visuose IS būsenose mažiausiai vienas iš esybių atributų turintis perskyrimo būtinumo apribojimą turi turėti reikšmę.
Naudojimas. Šis apribojimas naudojamas nebūtiniems esybių atributams.

Apribojimo pavyzdys. Esybės tipo objektas „Darbuotojas“ identifikuojamas pagal atributą „Numeris“ ir turi skiriamąjį būtinumo apribojimą neprivalomiems atributams „PasoNr“ ir „SocDraudimoNr“. Šie apribojimai renka semantiką, kad kiekvienas darbuotojas turi turėti paso numeri ar socialinio draudimo numeri, arba abu kartu. Kiekvienas iš šių dviejų atributų yra neprivalomas.



6 pav. Skiriamojo būtinumo apribojimas

- **Unikalumo apribojimai**

Unikalumo apribojimai pažymi, kokios atributų reikšmės imant bet kurias iš dviejų esybių rinkinio yra skirtingos.

Naudojimas. Šis apribojimas naudojamas esybės atributui ar atributų grupei.

Apribojimo pavyzdys. Esybės tipo objektas „Šalis“ identifikuojamas pagal „SaliesKodas“ reikalauja unikalumo apribojimo atributui „Vardas“ jei mes norime pažymėti, kad negali egzistuoti šalys su tuo pačiu pavadinimu. Unikalumas gali būti ne tik vieno atributo reikšmė, bet ir keleto atributų reikšmių kombinacijų.

Unikalumo apribojimai atributui (a) ir atributų grupei (b).

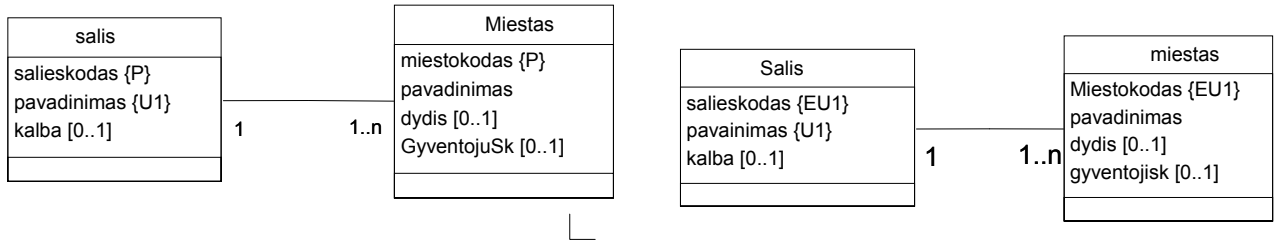
7 pav. Unikalumo apribojimai

- **Išorinio unikalumo apribojimai**

Šis apribojimas gali būti laisvai apibūdintas kaip unikalumo apribojimas skirtingiems esybių atributams. Konceptualiai susietų atributų reikšmės turi būti skirtingos nuo kitų reikšmių konceptualiai susietų su atributų reikšmių rinkiniu.

Naudojimas. Šis apribojimų tipas naudojamas atributams iš skirtingų esybių.

Apribojimo pavyzdys. Esybės tipo objekto „Salis“ atributo „SaliesKodas“ reikšmė kartu su esybės „Miestas“ atributo „MiestoKodas“ reikšme turi unikaliai identifikuoti miestą. Šalies kodas paprastai identifikuoja šalį, bet ne visada. Kaip bebūtų, šalies kodas ir miesto kodas unikaliai identifikuoja miestą. Ši instancija konceptualiai susieja atributų reikšmes iš skirtingų esybių.



a **b**

8 pav. Išorinio unikalumo apribojimai

- **Atributo reikšmės apribojimas**

Naudojimas. Šis apribojimas naudojamas atributams.

Apribojimo pavyzdys. Atributas lytis gali turėti dvi reikšmes: „V“ jei asmuo yra vyriškos giminės ir „M“ jei asmuo moteriškos giminės. Egzamino rezultatai turi gauti teigiama reikšmę nuo 1 iki 10 intervale.

a **b**

Enumeracijos modeliavimo pvz (a) ir reikšmių intervalo pvz (b).

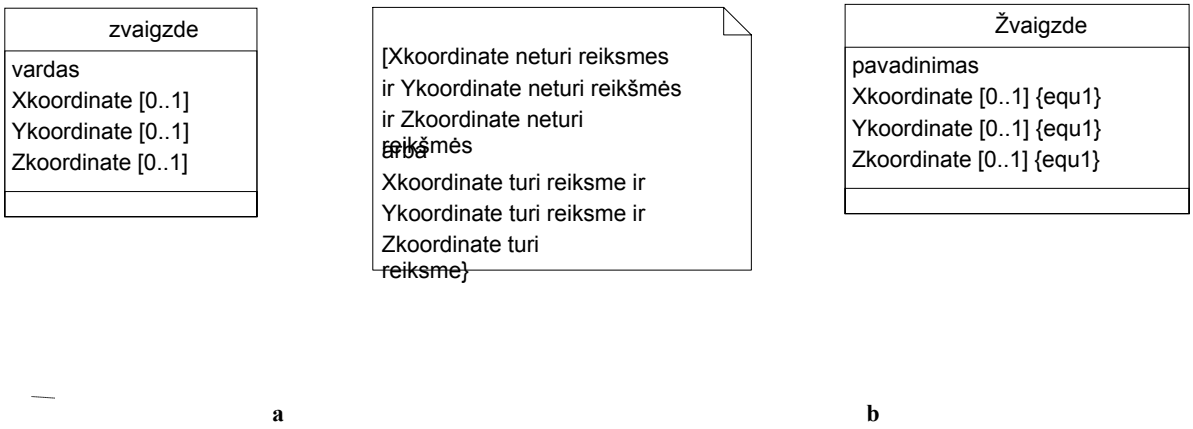
9 pav. Atributo reikšmės apribojimas

- **Lygybės apribojimai**

Lygybės apribojimai tarp esybės ryšių apibūdina lygia priklausomybę tarp vieno ar daugiau ryšių. Tai reiškia, kad jei esybė susijusi su vienu ryšiu, tai ji turi būti susijusi ir su visais kitais ryšiais bei negali egzistuoti reikšme susijusi tik su vienu ryšiu.

Naudojimas. Šis apribojimas naudojamas neprivalomiems esybių atributams ar neprivalomiems ryšiams.

Apribojimo pavyzdys. Nustatant žvaigždės pozicija danguje, nustatomos trys koordinatės: x, y, z arba nė viena iš jų. Šioje situacijoje esybės „Zvaigzde“ atributai yra „Xkoordinate“, „Ykoordinate“ ir „Zkoordinate“ turi lygumo apribojimus.

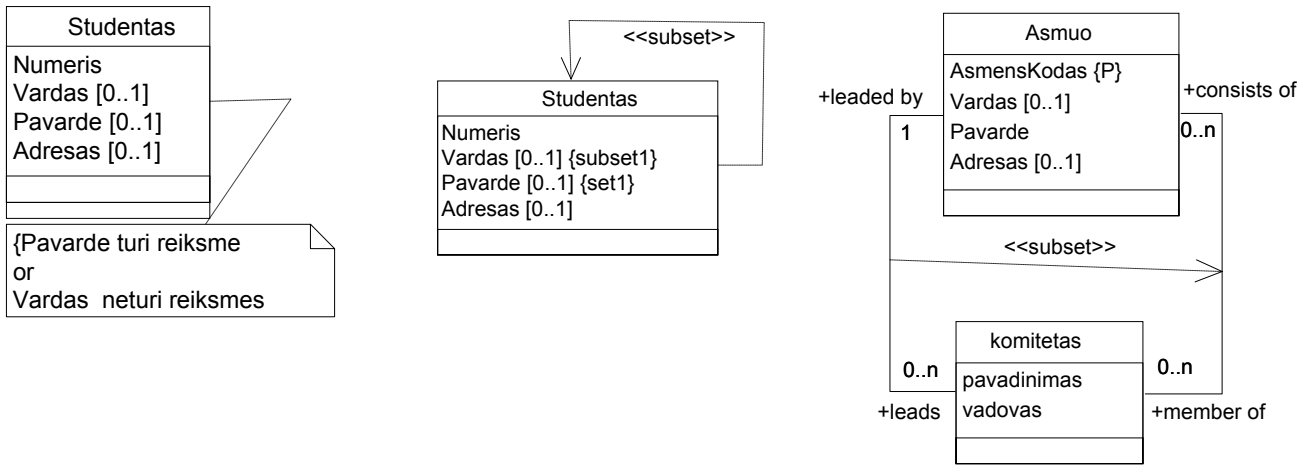


10 pav. Lygybės apribojimas

- **Poaibio apribojimas**

Naudojimas. Šis apribojimas naudojamas nebūtiniems esybių atributams ir ryšiams.

Apribojimo pavyzdys.



a **b** **c**

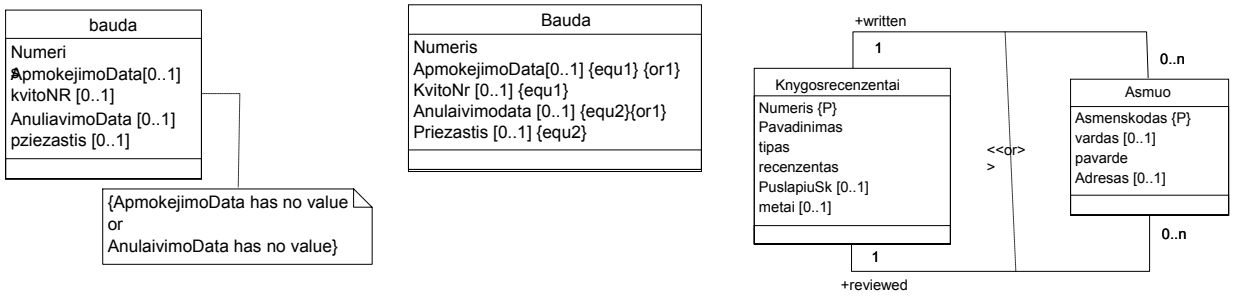
Poaibio apribojimo žymėjimas pastaboje objektui pavyzdys (a) ir poaibio apribojimo klasės egzemplioriui pavyzdys (b) ir poaibio apribojimo asociacijoms žymėjimas specialiu apribojimu pavyzdys (c).

11 pav. Poaibio apribojimas

- **Išskirtinumo apribojimas**

Naudojimas. Šis apribojimas naudojamas nebūtiniems esybių atributams ir ryšiams.

Apribojimo pavyzdys.

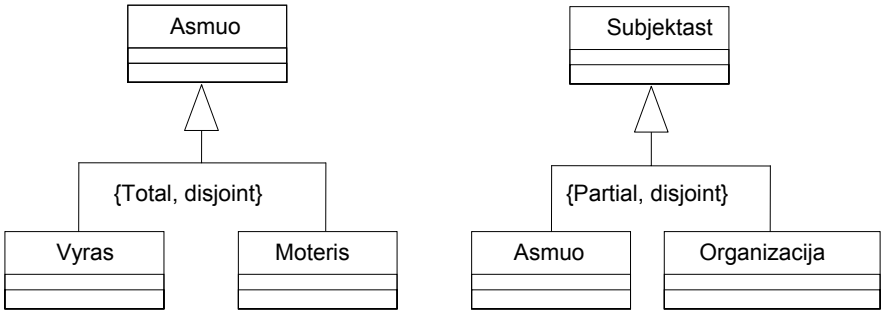


a **b** **c**

Išskirtinumo apribojimo pažymėjimas pastaboje objektui pavyzdys (a) ir išskirtinumo apribojimo atributams žymėjimas specialiu apribojimu pavyzdys (b) ir išskirtinumo apribojimo asociacijoms žymėjimas specialiu apribojimu pavyzdys (c).

12 pav. Išskirtinumo apribojimas

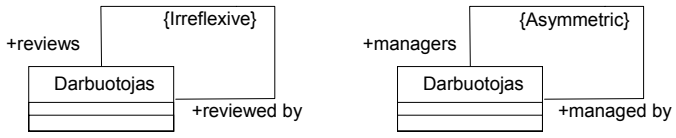
- **Išskirtinumo ir suskaidymo apribojimas**

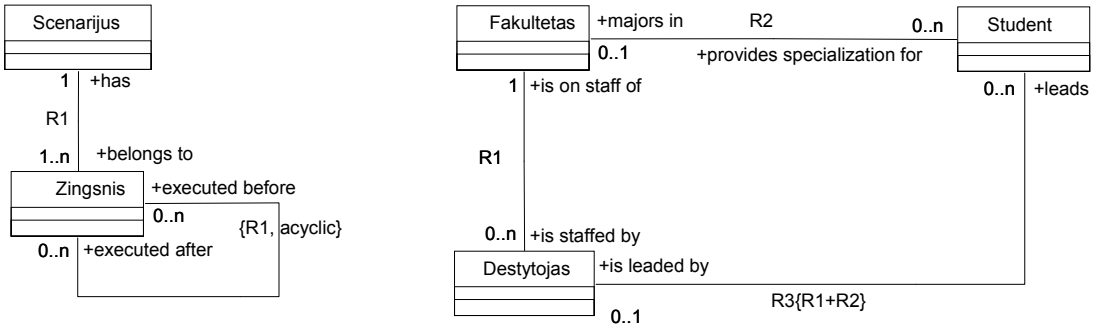


Išskirtinumo ir suskaidymo apribojimo žymėjimo UML klasių diagramoje pavyzdys.

13 pav. Išskirtinumo ir suskaidymo apribojimas

- **Refleksyvių ryšių apribojimai**





a **b**

Egzempliorių susiejimo ryšiais apribojimo refleksyviajam ryšiui (a) ir asociacijai pavyzdys (b).

14 pav. Refleksyvių ryšių apribojimai

3.5. Apribojimų realizavimo būdų DBVS analizė

Paprasčiausia ir dažniausiai naudojama apribojimų (constraints) realizavimo idėja yra naudoti išorinius raktus ir CHECK funkcijas, tačiau šių galimybių nepakanka.

Apribojimų realizavimui pasirinkta ORACLE 10g duomenų bazių valdymo sistema pilnai atitinkanti daugelį reikalavimų. ORACLE 10g DBVS kai kuriuos paprastus apribojimus palaiko automatizuotai: pirminis atributo identifikatorius, atributo identifikatorius, nurodomasis atributo apribojimas, būtinasis atributo apribojimas, vidinis unikalumo apribojimas. Kitų likusių apribojimų realizavimui panaudosime DBVS palaikančiomis CHECK funkcijomis, TRIGERIAIS ir PAKETAIS.

CHECK (condition)

CREATE TRIGGER

CREATE PACKAGE

CREATE PACKAGE (Paketo sukūrimas) tai duomenų bazės objektas logiškai grupuoja duomenų bazės objektus, PL/SQL tipus paprogrames. Šiuo atveju jis naudojamas sisteminio masyvo sukūrimui, skirto įrašų eilutėms identifikuoti.

Daugelis duomenų bazių gamintojų naudoją mechanizmą apribojimų vykdymui, pavadintą trigeriais. Viena iš problemų susijusi su apribojimais yra tai, kad jie turi būti patikrinti esant kiekvienam duomenų bazės pakitimui, nebent DBVS yra pakankamai intelektuali, kad žinoti kada šitie teiginiai turėtų būti patikrinti.

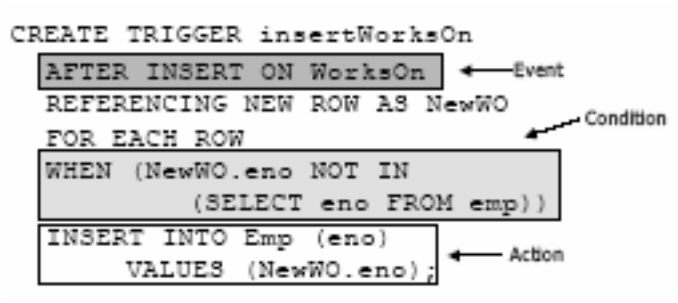
TRIGERIAI

Tokios problemos sprendimas aiškiai nurodyti DBVS, kurie įvykiai yra suinteresuoti ir pateikti koks veiksmas turi būti vykdomas kada įvykis įvyksta. Trigeriai yra naudojami atlikti reikiamus veiksmus kada nurodytas įvykis įvyksta. Jie yra naujai standartizuoti SQL3.

Trigeriai taip pat vadinami event-condition-action(ECA) taisyklėmis. *Event(ivykis)* sąlygoja taisyklės veikimo pradžią. *Condition(Sąlyga)* tikrinama taisyklės veikimo pradžioje ir Action(*Veiksmas)* yra vykdomas kai įvyksta *EVENT* ir CONDITION yra teisinga.

Apžvelkime situacija kada trigeriai yra naudingi:

The worksOn ryšys turi antrini rakta i Emp(eno). Jei vartotojas įrašo nauja įrašą i WorksOn lentelę ir employee neegzistuoja, tokiu atveju naujo įrašo įterpimas neįvyksta. Kaip bebūtų naudojant trigerius, mes galime leisti įterpti įrašą į WorksOn ir tada sukurti naują įrašą Emp lentelėje taip nepažeidžiant antrinio rakto apribojimų.



Trigerių sintaksė

Nurodoma sąlyga leidžia priskirti vardus eilutei ar lentelei susijusiai su trigerio iššauktu įvykiu.
INSERT leidžia pateikti nauja įrašą (eilutės lygio) ar naują rinkini įrašų (statement-level)
DELETE leidžia pateikti seną įrašą ar lentelę
UPDATE leidžia abu ir seną ir naują

Šie įrašai ar lentelės gali būti priskiriamos naudojant sintaksę

[NEW OLD] [TUPLE TABLE] AS <name>

Pvz: Statement-level triggeris atnaujinant duomenis:
REFERENCES OLD TABLE AS oldTbl
NEW TABLE as newTbl

Condition(sąlyga) yra naudojama išsiaiškinti, kada trigerio veiksmas atliekamas.
Bet kokia SQL Boolean tipo salyga gali būti naudojama. Sąlygos sakinys yra vykdomas prieš arba po trigerio įvykio, priklausomai nuo to kokias BEFORE ar AFTER salygas naudoja. Veiksmas įvykdomas kai įvykis įvyksta ir sąlyga yra patenkinama. Veiksmas gali būti vienas ar daugiau SQL sakinių.
Jei daugiau nei vienas SQL sakinys bus vykdomas, jie turi būti BEGIN….END bloke.

Trigerio Event (įvykio) pagrindinis vykdymas detaliai nusakytas pačio trigerio apibrėžime. Mūsų atveju, event yra UPDATE, DELETE, INSERT konstatavimas taikomas duomenų bazės lentelei. Laisvai pasirenkamas stulpelių sąrašas gali būti nusakytas tokiu atveju, toliau uždrausti rinkinį atnaujinimo įvykių, kurie sužadino trigerį.

Trigeris taip pat turi sužadinimo laiką, kuris apibūdina jei trigeris yra vykdomas prieš (before) arba po (after) įvykio ir „susmulkinimo“ (granularity), kuris apibūdina kiek kartų trigeris yra vykdomas esamam įvykiui. Before-triggers vykdomas prieš tam tikrą apibrėžtą įvykį ir yra labai naudingas sąlygojant įvedamus

duomenis prieš pakeitimus , kurie yra taikomi duomenų bazei ir nustatyti svarbūs apribojimai. After-trigger vykdo po savo įvykio ir yra paprastai naudojami įterpimo taikymo logikai, kuri paprastai kai pakeitimai pasibaigia. Trigerio „susmulkinimas“ apibrėžiamas, kaip bet kuris FOR EACH ROW arba FOR EACH STATEMENT, remiantis kaip row-level(eilutės lygio) ir statement-level trigerių nurodyta tvarka. Kai row-level trigerio įvykis įvyksta, trigeris yra vykdomas kartą kiekvienai eilutei, kuriai yra taikomas šis įvykis. Jei nė viena eilutė nesužadinta ar nepaveikta, tokiu atveju trigeris niekada neįvertinamas. Tačiau statement-level trigeris yra vykdomas tiksliai vieną kartą, neatsižvelgiant į tai ar įvykis įvyko, net ir jei įvykis nepakeitė jokių eilučių. Kaip ir apribojimai, abu row-level ir statement-level after-triggerrs privalo vykdyti tik kai trigerio after įvykis baigia savo vykdymą; before-triggers privalo vykdyti tiktai prieš tai kai trigerių įvykių pakitimai yra pritaikomi. Reiktų pažymėti ir tai, kad „susmulkinimas“ (granularity) nenurodo, kada trigeris yra vykdomas. Aišku, kad yra daug sąlygų, kada row-level trigeriai gali būti saugiai ir optimaliai apdorojami. Pvz. jei row-level update trigeris lentelėje T gauna informaciją apie stulpelio vidurkį lentelėje T , tada vidurkis negali būti skaičiuojamas viduryje trigerio atnaujinimo, vidurkis tokiu atveju turi būti skaičiuojamas tiktais prieš arba po bet kokių pakitimu lentelėje T. Kiekvienas trigeris turi priėjimą prie before-transition(prieš-perėjimo) reikšmių ir after-transition (po-perėjimo) reikšmių, dėka apibūdinimo perėjimo kintamųjų (transition variables) ir perėjimo lentelių (transition tables). Apibūdinimas „referencing NEW as N“ apibūdina N kaip vieną eilutę koreliacijos kintamąjį, kuris sulaiko/saugo eilutės reikšmę duomenų bazėje, nedelsiant kai įvyksta pakitimai. Panašiai, „referencing OLD as 0“ apibūdina 0 kaip viena eilutę koreliacijos kintamąjį, sulaikantį tos pačios eilutės reikšmę duomenų bazėje dar prieš tai kai įvykdomi pakitimai. Jeigu abu OLD ir NEW perėjimo kintamieji yra aprašomi kaip, kad aukščiau, tada nauja ir sena reikšmės gali būti sulyginamos. Pvz. jeigu trigerio lentelė yra EMP ir EMP turi stulpelį atlyginimas, tada mes galime patikrinti ar pakitimas buvo įvykdytas, sulyginant 0.atlyginimas su N.atlyginimas. Perėjimo lentelės yra taip pat nurodomos naudojant raktinius rodžius NEW_TABLE ir OLD_TABLE. Jeigu NT aprašysime kaip naują perėjimo lentelę, tai ji bus virtuali lentelė užlaikanti visas modifikuotas reikšmes po atnaujinimo baigimo.
Ne visi perėjimo kintamųjų ir perėjimo lentelių tipai yra galimi kiekvienam trigerio tipui. Trigeriai, sukurti įterpimo įvykiams gali matyti tik naujas reikšmes ir trigeriai sukurti šalinimo įvykiams, gali matyti tik senas reikšmes. Kai atnaujinimo įvykiams sukurti trigeriai gali matyti abi seną ir naują reikšmes.

Bet kuris trigeris turi action, kuris nebūtinai yra reaguoja į condition; action yra vykdomas tuo atveju, kai trigerio event (įvykis) įvyksta ir jei nurodyta condition (sąlyga) yra patenkinama. Condition yra viena, neuždrausta paieškos sąlyga, o action – procedūra, turinti seką SQL sakinių. Abu action ir condition gali užklausti perėjimo kintamuosius, bei lenteles taip kaip esamą reikšmę duomenų bazėje. Taip pat šios užklausos gali iškviesti vartotojo apibrėžtas funkcijas.

Nuo to laiko kai before-trigger pasirodo vykdyti vien tik prieš įvykiui įvykstant, bet kokios duomenų bazės užklausos, turinčios conditions ar actions turi nuskaityti duomenų bazės būsenas, prieš bet kokias modifikacijas atliekamas šio event(įvykio). Panašiai, condition ir action visų after-trigger privalo nuskaityti duomenų bazės būsenas, po to kai modifikacijos įvykdomos. Jei after-trigger reikia užklausti duomenų bazės būklės reikalingos įvykiui, tada tai gali rekonstruoti naudojant perėjimo lenteles. Trigerio action yra atominė procedūra pvz. SQL Procedural Stored Module (PSM) gali turėti SQL sakinius, sujungtus su kitais procedūriniais konstruktoriais. SQL sakiniai yra vykdomi eilės tvarka, kurioje jie įvyksta procedūros nustatyme. Jei bent vienas SQL sakinys nepavyksta, visas trigerio action (veiksmas) yra grįžtamas į pradžią, įtraukiant ir trigerį iškviečianti veiksmą. Trigeriai gali netiesiogiai sukelti statement-level roll-back, sukeliant klaidą, pvz. naudojant SQL signalo konstatavimą. Kada tai atsitinka, susidedantis SQL sakinys nepavyksta , ir kaip aukščiau apibūdinta, visas trigerio veiksmas yra rolled-back (atsukamas atgal), įskaitant ir įvykį, kuris sužadino trigerį.

Kai kurie trigeriai gali turėti tą patį įvykio ir aktyvavimo laiką. Taigi, sudėtiniai trigeriai gali būti tuo pat metu tinkami vykdymui. Kada taip atsitinka tinkami trigeriai privalo būti vykdomi remiantis kai kuriais pastebimais visapusiškos tvarkos nurodymais ištisam trigerių rinkiniui.

3.6. Analizės išvados

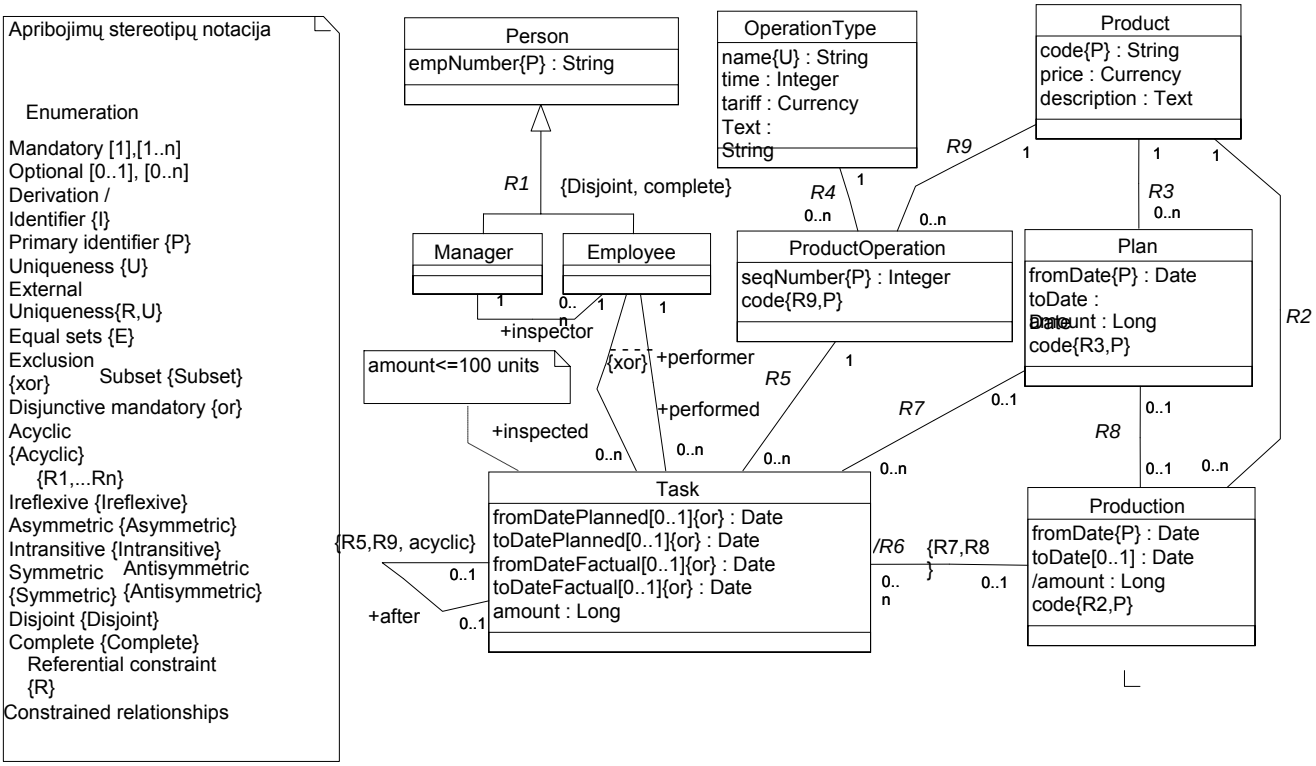
- Palyginti įvairiuose modeliavimo metoduose (ER, EER, HERM, ORM, UML, xUML, RDBMS) naudojami apribojimai.
- Išanalizuotos apribojimų specifikacijos ir jų realizavimo galimybės.
- Pateikti konceptualūs apribojimų modeliai.
- Apribojimų realizavimui pasirinkta ORACLE 10g DBVS.
- Išnagrinėti ir aprašyti ORACLE trigerių sintaksė, bei veikimo principai.

4. Apribojimų realizavimo metodika

Šiame skyriuje pateikiami įvairių tipų apribojimų realizavimo modeliai, kurių iliustracijai naudojamas sudėtingas duomenų modelis, turintis įvairių tipų apribojimų.

4.1. Duomenų modelio pavyzdys

Apribojimų realizavimui tirti paimtas sudėtingas duomenų modelis, aprašytas 15 paveiksle:

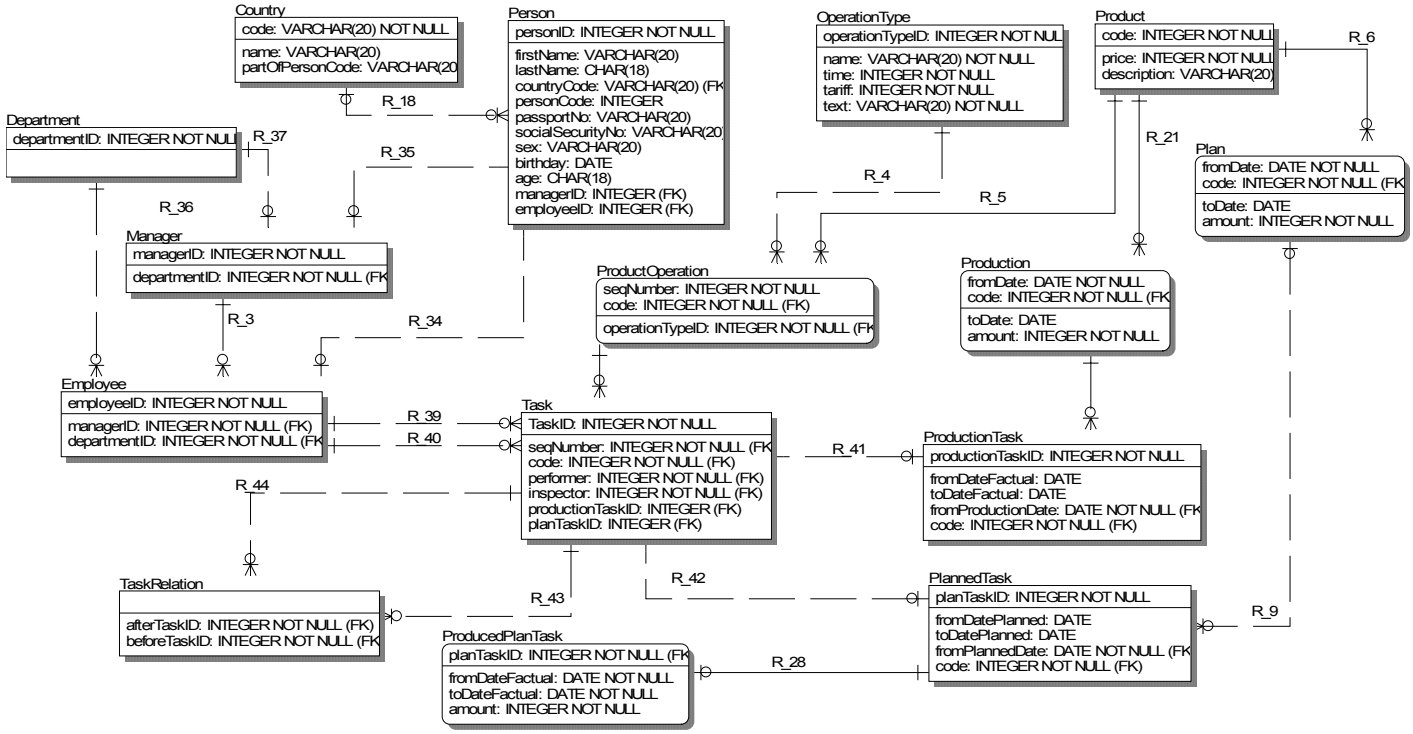


15 pav. Gamybos sistemos duomenų modelis

Gamybos sistemoje 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. Kadangi modelis apima daug įvairių variantų, jame gausu įvairių apribojimų, kurie pažymėti stereotipais, parodytais paveikslo kairėje.

4.2. Duomenų bazės modelis

Tiriant apribojimų realizavimą, iš duomenų modelio sudaryta duomenų bazės schema pateikta 16 paveiksle:



16 pav. Duomenų bazės modelis

4.3. Apribojimų įgyvendinimo duomenų bazėje modeliai

Iš viso galima išskirti keliolika apribojimų tipų, vieni realizuojami kuriant trigerius, check funkcijas, kiti yra tiesiog automatizuoti pačiose duomenų bazių valdymo sistemose. Pateiksime apribojimų realizavimo galimus sprendimo būdus.

4.3.1. Patys paprasčiausi apribojimai

Pirminis atributo identifikatorius, atributo identifikatorius, nurodomasis atributo apribojimas, būtinasis atributo apribojimas, skiriamasis būtinumo apribojimas. Šių apribojimų charakteristika pateikta 17 paveikslo lentelėje.

Apribojimas	**Naudojimas**	**Realizavimo galimybės**	**Pasirinktas realizavimo metodas**	**Atitikmuo DBVS**
Pirminis atributo indentifikatorius: *(arba grupės atributų).(***Primary identifier on attribute**)	Pirminis atributo identifikatorius yra naudojamas lentelės lauko raktui nusakyti	Per SQL sintaksę, Automatiškai užtiktinami DBVS	Naudojant SQL sintaksę	PRIMARY KEY
Atributo indentifikatorius: *(arba grupės atributų).* (**Identifier constraint on attribute**)	Atributo identifikatorius yra naudojamas lentelės lauko unikalumui (unique) nusakyti.	Per SQL sintaksę, Automatiškai užtiktinami DBVS	Naudojant SQL sintaksę	name varchar(20) UNIQUE UNIQUE (a,b)
Nurodomasis atributo apribojimas: **(Referential constraint on attribute)**	Šis apribojimas DBVS yra naudojamas norint susieti lenteles tarpusavio ryšiu. T.y. šio proceso metu yra susiejamas lentelės laukas, turintis pirminio atributo apribojimą, ir lentelės laukas, kuriame nusakytas nurodomojo atributo apribojimas.	Per SQL sintaksę, Automatiškai užtiktinami DBVS	Naudojant SQL sintaksę	FOREIGN KEY (<lauko_pav REFERENCING <lentelės_pav>)
Būtinasis atributo apribojimas **(Mandatory constraint on attribute)**	Lentelės kūrimo procese nustatomas būtinas apribojimas laukui, kuris privalo turėti kažkokią reikšmę .	Per SQL sintaksę, Automatiškai užtiktinami DBVS	Naudojant SQL sintaksę	< lauko_pav > NOT NULL
Skiriamasis būtinumo apribojimas: **(Disjunctive mandatory constraint)**	Šis skiriamasis būtinumo apribojimas naudojamas siekiant patikrinti vienokio ar kitokio lauko reikšmę:	Naudojant CHECK funkciją	Naudojant CHECK funkciją	CHECK (<lauko_pav> IS NOT NULL OR <lauko_pav> IS NOT NULL … OR <lauko_pav> IS NOT NULL);

17 pav. Lentelė

Pirminis atributo identifikatorius: (arba grupės atributų). (Primary identifier on attribute)

Naudojimas: lentelės lauko raktui nusakyti.

Pirminio identifikatoriaus naudojimas, lentelės stulpeliams ORACLE 10g sintaksėje:

ALTER TABLE <lenteles_vardas>
ADD CONSTRAINT <pirminio_rakto_apribojimo_vardas>
PRIMARY KEY (<stulpelio_vardas, stulpelio_vardas, ...>);

Norint panaudoti šį pirminio atributo apribojimo šabloną, reikia įrašyti pasirinktos lentelės pavadinimą, norimą apribojimo pavadinimą ir kokiam lentelės stulpeliui ar laukus jis bus naudojamas.

Atributo identifikatorius: (arba grupės atributų). (Identifier constraint on attribute)

Naudojimas: atributo identifikatorius yra naudojamas lentelės lauko unikalumui (unique) nusakyti.

Atributo identifikatoriaus naudojimas, lentelės stulpeliams ORACLE 10g sintaksėje:

ALTER TABLE <lenteles_vardas>
ADD CONSTRAINT <apribojimo_pav>
UNIQUE (<lauko_pav, lauko_pav>)

Norint panaudoti šį atributo unikalumo apribojimo šabloną, reikia įrašyti pasirinktos lentelės pavadinimą, norimą apribojimo pavadinimą ir kokiam lentelės stulpeliui ar laukus jis bus naudojamas.

Nurodomasis atributo apribojimas: (Referential constraint on attribute)

Naudojimas: šis apribojimas DBVS yra naudojamas norint susieti lenteles tarpusavio ryšiu. T.y. šio proceso metu yra susiejamas lentelės laukas, turintis pirminio atributo apribojimą, ir lentelės laukas, kuriame nusakytas nurodomojo atributo apribojimas.

Nurodomojo atributo apribojimo naudojimas, lentelės stulpeliams ORACLE 10g sintaksėje:

ALTER TABLE < lenteles_vardas >
ADD CONSTRAINT < apribojimo_pav >
FOREIGN KEY (<column_name, column_name, ...>)
REFERENCING <table_name> (<column_name, column_name, ...>)

Norint panaudoti šį nurodomąjį atributo apribojimo šabloną, reikia įrašyti pasirinktos lentelės pavadinimą, norimą apribojimo pavadinimą ir kokiam lentelės stulpeliui ar laukus jis bus naudojamas bei su kokia lentele ir lauku jis bus susiejamas.

Būtinasis atributo apribojimas (Mandatory constraint on attribute)

Naudojamas: lentelės kūrimo procese nustatomas būtinas apribojimas laukui, kuris privalo turėti kažkokią reikšmę .

Naudojimas, lentelės stulpeliams ORACLE 10g sintaksėje:

```
CREATE TABLE <lenteles_pav> (
  lauko_pav datatype [būtinasis atributo apribojimas lauko_pav]
  [,lauko_pav datatype [būtinasis atributo apribojimas lauko_pav]]
);
```

Norint panaudoti šį būtinojo atributo apribojimo šabloną, reikia įrašyti pasirinktos lentelės pavadinimą, kokiam ar kokiems lentelės stulpeliui ar laukui jis bus naudojamas.

Skiriamasis būtinumo apribojimas: (Disjunctive mandatory constraint)

Naudojimas: šis skiriamasis būtinumo apribojimas naudojamas siekiant patikrinti vienokio ar kitokio lauko reikšmę.

Norint panaudoti šį nurodomąjį atributo apribojimo šabloną, reikia įrašyti pasirinktos lentelės pavadinimą, norimą apribojimo pavadinimą ir kuriems laukams norėsite pritaikyti šį apribojimą.

```
ALTER TABLE <lenteles_pav>
ADD CONSTRAINT <apribojimo_pav>
CHECK (<lauko_pav> IS NOT NULL OR <lauko_pav> IS NOT NULL ... OR <lauko_pav> IS NOT NULL);
```

4.3.2 Unikalumo apribojimai

Unikalumo apribojimai yra išskirti du: vidinis ir išorinis apribojimai pavaizduoti 18 paveikslo lentelėje.
Vidinio unikalumo apribojimą praktiškai automatizuoją ORACLE 10g duomenų bazių valdymo sistema.
Išorinio unikalumo apribojimo realizavimui reikia sukurti du vaizdus (view) ir du trigerius.

Apribojimas	**Naudojimas**	**Realizavimo galimybės**	**Pasirinktas realizavimo metodas**	**Atitikmuo DBVS**
***Vidinis unikalumo apribojimas (*Internal uniqueness constraint)*:**	Vidinis atributo apribojimas yra naudojamas lentelės lauko unikalumui nusakyti	Per SQL sintaksę, Automatiškai užtiktinami DBVS	Naudojant SQL sintaksę	ADD CONSTRAINT <apribojimo_pav> UNIQUE (<lauko_pav, lauko_pav, ...>);

Išorinis unikalumo apribojimas: **(External uniqueness constraint)**	Kai reikia identifikuoti keliose lentelėse susietus egzistuojančius atributus ir nustatyti unikalumą	Realizacijai naudojami vaizdai (view) lentelėse, kuriose šis apribojimas bus taikomas ir specialiai vaizdams (view) naudojamus trigerius INSTEAD OF	Kuriant view ir jiems skirtus trigerius INSTEAD OF	Duomenų bazėje sukurti du vaizdai view tarp ryšiu susietų lentelių, bei atitinkami jiems du trigeriai.
Atributo reikšmės apribojimas **(Value constraint on attribute)**	Uždraudžiant atributo reikšmes baigiamojoje reikšmių nurodymo stadijoje, įvertinant pradines ir galutines reikšmes (intervalą). - Kai leidžiamos reikšmės yra įvardintos - Kada leidžiamos reikšmės yra apibrėžtos nurodant intervalą - Kada leidžiamos reikšmės yra apibrėžtos nurodant intervalą ar įvardintos,	Naudojant CHECK	Naudojant CHECK funkcijas	CHECK (<stulpelio_pav > = <reiksme> OR < stulpelio_pav > = <reiksme >); CHECK (<column_name> BETWEEN <value> AND <value> ... OR <column_name> BETWEEN <value> AND <value>) CHECK (<stulpelio_pav > = <reiksme> OR < stulpelio_pav > BETWEEN <reiskme> AND <reiskme> ... OR < stulpelio_pav > BETWEEN <reiksme> AND <reiksme>));
Apribojimas	**Naudojimas**	**Realizavimo galimybės**	**Pasirinktas realizavimo metodas**	**Atitikmuo DBVS**
Įgytas atributo apribojimas.**(Derived attribute constraint)**	Jei gautas atributas yra skaičiuojamas iš atributų, kurie priklauso tai pačiai lentelei, tokiu atveju gauto atributo skaičiavimo taisyklė gali būti vykdoma trigeryje. Prieš INSERT ar UPDATE kiekvieno lentelės įrašo bus paruošiama nauja gauto atributo reikšmė	Sukurto trigerio pagalba	Trigeriu	Nera

18 pav. Lentelė

Vidinis unikalumo apribojimas (Internal uniqueness constraint)*:*

Naudojimas: Vidinis atributo unikalumo apribojimas yra naudojamas lentelės lauko unikalumui nusakyti.

Atributo unikalumo naudojimas, lentelės stulpeliams ORACLE 10g sintaksėje:

Šablonas ORACLE sintaksėje:

ALTER TABLE <lenteles_pav>
ADD CONSTRAINT <apribojimo_pav>
UNIQUE (<lauko_pav, lauko_pav, ...>);
Išorinis unikalumo apribojimas: (External uniqueness constraint)

Naudojimas: Kai reikia identifikuoti keliose lentelėse susietus egzistuojančius atributus.

Atliksime išorinio unikalumo apribojimo realizaciją. Išorinio unikalumo apribojimo realizacijai atlikti naudojami vaizdai (view) lentelėse, kuriose šis apribojimas bus taikomas ir su jomis susieti specialūs vaizdams taikomi trigeriai INSTEAD OF.

Vaizdo sukūrimo šablonas:

```
CREATE OR REPLACE VIEW <vaizdo_pav> AS
SELECT <lauko_pav, lauko_pav,…> FROM <lenteles_pav>;
```

Trigerio INSTEAD OF kūrimas:

```
CREATE OR REPLACE TRIGGER <trigerio_pav>
  INSTEAD OF UPDATE ON <View_pav>
  FOR EACH ROW
DECLARE
  a_Exists NUMBER := 0;
  Invalid_record EXCEPTION;
BEGIN
  SELECT count(p.lauko_pav) INTO a_Exists
  FROM <lenteles_pav> p, <lenteles_pav1> c
  WHERE p.lauko_pav = c.code
  and c.<lauko_pav> = :new.<lauko_pav>
  and p.<lauko_pav> IN (SELECT p.<lauko_pav>
                        FROM lenteles_pav p, lenteles_pav1 c
                        WHERE p.<lauko_pav> = c. <lauko_pav>
                        and c. <lauko_pav>  = :old. <lauko_pav>);
  IF a_Exists > 0 THEN
    RAISE Invalid_record;
  ELSE
    UPDATE country c set
    c.<lauko_pav> = :new.<lauko_pav>,
    c.<lauko_pav> = :new.<lauko_pav>,
    c.<lauko_pav> = :new.<lauko_pav>
    where c.<lauko_pav> = :old.<lauko_pav>;
  END IF;
  EXCEPTION
     WHEN Invalid_record THEN
       RAISE_APPLICATION_ERROR (
         num=> -20107,
         msg=> 'Pranesimas');
END <trigerio_pav>;
```

Atributo reikšmės apribojimas (Value constraint on attribute)

Naudojimas: Uždraudžiant atributo reikšmes baigiamojoje reikšmių nurodymo stadijoje, įvertinant pradines ir galutines reikšmes (intervalą).

Šis apribojimas realizuojamas naudojant CHECK funkciją.

Galima išskirti tokius tris naudojimo tipus: įvardinant reikšmes, nurodant intervalą ir naudojami abu kartu.

Pateiktas reikšmės apribojimas, kada leidžiamos reikšmės yra įvardintos, panaudojant check apribojimą:

ALTER TABLE <lenteles_pav>
ADD CONSTRAINT <apribojimo_pav>
CHECK (<stulpelio_pav > = <reiksme> OR < stulpelio_pav > = <reiksme > ... OR < stulpelio_pav > = <reiksme >);

Pateiktas reikšmės apribojimas, kada leidžiamos reikšmės yra apibrėžtos nurodant intervalą, panaudojant check apribojimą:

ALTER TABLE < lenteles_pav >
ADD CONSTRAINT < apribojimo_pav >
CHECK (<stulpelio_pav> BETWEEN <reiksme> AND <reiskme> ... OR <stulpelio_pav> BETWEEN <reiksme> AND <reiksme>);

Pateiktas reikšmės apribojimas, kada leidžiamos reikšmės yra apibrėžtos nurodant intervalą ar įvardintos, panaudojant check apribojimą:

ALTER TABLE < lenteles_pav >
ADD CONSTRAINT < apribojimo_pav >
CHECK (<stulpelio_pav > = <reiksme> OR < stulpelio_pav > = <reiksme> ... OR < stulpelio_pav > BETWEEN <reiskme> AND <reiskme> ... OR < stulpelio_pav > BETWEEN <reiksme> AND <reiksme>));

Išvestinio atributo apribojimas (Derived attribute constraint)

Naudojimas: Jei gautas atributas yra skaičiuojamas iš atributų, kurie priklauso tai pačiai lentelei, tokiu atveju gauto atributo skaičiavimo taisyklė gali būti vykdoma trigeryje. Prieš INSERT ar UPDATE kiekvieno lentelės įrašo bus paruošiama nauja gauto atributo reikšmė.

```
CREATE OR REPLACE TRIGGER derivedAttribute
  BEFORE INSERT OR UPDATE OF <lauko_pav> ON <lenteles_pav>
  for each row
BEGIN
  “Norimas atlikti veiksmas PL/SQL kalba”
END derivedAttribute;
```

4.3.3Vientisumo apribojimai ryšiams (Integrity constraints on relationship or relationships)

Išskirti trys vientisumo apribojimų ryšiams tipai: daugialypiškumo, apibendrininimo/specifikavimo ir refleksyvių ryšių apribojimai. Šie apribojimai pateikti 19 paveikslo lentelėje.

Apribojimas	**Naudojimas**	**Realizavimo galimybės**	**Pasirinktas realizavimo metodas**	**Atitikmuo DBVS**
***Daugialypiškumo apribojimas* (Multiplicity constraints):**	Daugialypiškumo apribojimas, kuris uždraudžia lentelės įrašų kiekį, susijusių su lentelės vienu įrašu esant didesnėm ar mažesnėm ribom.	Sukurto trigerio pagalba, sukurto vaizdo (view) pagalba	Trigeriu ir view	Nėra
***Apibendrinimo/specifikavimo ryšių apribojimai* (Constraints on generalization/specialization relationship).**	CHECK apribribojimo *išskirstymo* apribojimui dviems papildomiems stulpeliams sukurtiems pagrindinėje lentelėje CHECK apribribojimo *pilnam* apribojimui dviems papildomiems stulpeliams sukurtiems pagrindinėje lentelėje CHECK apribribojimo *pilnam* ir *išskirtymo* apribojimui dviems papildomiems stulpeliams sukurtiems pagrindinėje lentelėje:	Naudojant CHECK apribojimų funkcijas	Naudojant CHECK funkcijas	Nėra

Apribojimas	**Naudojimas**	**Realizavimo galimybės**	**Pasirinktas realizavimo metodas**	**Atitikmuo DBVS**
***Refleksyvių ryšių apribojimai:* (Constraints on reflexive associations)**	Yra šeši apribojimų tipai refleksyviems ryšiams. Irreflexive, Asymmetric, Acyclic, Intransitive, Symmetric, Antisymmetric	Visų tipų realizacija imanoma panaudojus tris trigerius. Pirmi du trigeriai visiems tipams vienodi. Skiriasi tik pagrindinis trečias trigeris.	Trimis trigeriais	Nėra

19 pav. Lentelė

Daugialypiškumo apribojimas (Multiplicity constraints):

Naudojimas: daugialypiškumo apribojimas, kuris uždraudžia lentelės įrašų kiekį, susijusių su kitos lentelės vienu įrašu esant didesnėm ar mažesnėm ribom, gali būti įgyvendinamas naudojant trigeriu.

```
CREATE OR REPLACE TRIGGER <trigerio_pav>
  AFTER INSERT OR UPDATE ON <lenteles_pav>
DECLARE
  a_Count NUMBER := 0;
  Invalid_record EXCEPTION;
BEGIN
  Select <lauko_pav>,(count(<lauko_pav>))
  from <lenteles_pav> B
  group by B.<lauko_pav>
  having count(b.<lauko_pav)>2 and count(b.<lauko_pav>)<5
  If (a_Count > 0) Then
    RAISE Invalid_record;
  END IF;
  EXCEPTION
     WHEN Invalid_record THEN
       RAISE_APPLICATION_ERROR (
         num=> -20107,
         msg=> 'Pranesimas');
     WHEN NO_DATA_FOUND THEN NULL;
END <trigerio_pav>;
```

Apibendrinimo/specifikavimo ryšių apribojimai **(Constraints on generalization/specialization relationship).**

Yra trys pagrindinės strategijos, kurios yra naudojamos įgyvendinant apibendrinimo/specifikavimo ryšius:

- Visos klasės hierarchiškai prijungtos prie vienos lentelės. Lentelė turi lauką, kuris reikalingas kaip lauko diskriminatorius, tai yra laukas, kurio reikšmes identifikuoja specifinė subklasė, kuriai eilutei priklauso.
- Kiekviena subklasė yra priskirta skirtingai lentelei. Kiekviena lentelė gauna lauką visiems atributams super tipo su originaliu pasirenkamumu.
- Hierarchijų super klasė, pateikiama viena lentele. Kiekviena subklasė yra pateikiama skirtinga lentele, kuri saugo atitinkamai svarbius subklasės laukus, taip kaip laukai yra pateikiami pirminio rakto pavidalu.

Nepaisant visų šitų variantų privalumų ir trūkumų, reikia pasirinkti, kokia strategija bus naudojama. Bet nei viena iš jų nėra gera, jei mes naudosime apribojimus apibendrinimo/specifikavimo ryšiams.

CHECK apribojimo *išskirstymo* apribojimui dviems papildomiems stulpeliams sukurtiems pagrindinėje lentelėje:

ALTER TABLE <lenteles_pav>
ADD CONSTRAINT <apribojimo_pav>
CHECK ((<stulpelio_pav> IS NOT NULL AND < stulpelio_pav > IS NULL) OR
(<stulpelio_pav > IS NULL AND < stulpelio_pav > IS NOT NULL) OR
(<stulpelio_pav > IS NULL AND < stulpelio_pav > IS NULL));

CHECK apribribojimo *pilnam* apribojimui dviems papildomiems stulpeliams sukurtiems pagrindinėje lentelėje:

ALTER TABLE < lenteles_pav >
ADD CONSTRAINT < apribojimo_pav >
CHECK (<stulpelio_pav > IS NOT NULL OR < stulpelio_pav > IS NOT NULL);

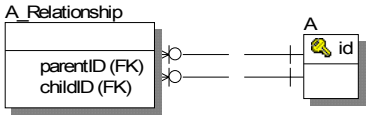
CHECK apribojimo *pilnam* ir *išskirstymo* apribojimui dviems papildomiems stulpeliams sukurtiems pagrindinėje lentelėje:

```
ALTER TABLE < lenteles_pav >
ADD CONSTRAINT < apribojimo_pav >
      CHECK ((<stulpelio_pav > IS NOT NULL AND < stulpelio_pav > IS NULL) OR
      (<stulpelio_pav > IS NULL AND < stulpelio_pav > IS NOT NULL));
```

Refleksyvių ryšių apribojimai: **(Constraints on reflexive associations)**

Yra šeši apribojimų tipai refleksyviems ryšiams.

Irreflexive, Asymmetric, Acyclic, Intransitive, Symmetric, Antisymmetric.



Paveikslėlyje matyti refleksinio ryšio pavyzdys. Reliaciniame modelyje galima sujungti skirtingais būdais. Panaudosime pagrindinę lentelę A ir viena papildomą lentelę A_Relationship ryšių saugojimui tarp lentelės A įrašų. Apribojimų vykdyme reikės naudoti check apribojimus arba trigerius, kad užtikrinti visas įmanomas atributų kombinacijas parentID ir childID.

“Irreflexive” apribojimas gali būti įgyvendinamas naudojant check apribojimus:

ALTER TABLE <lenteles_pav>
ADD CONSTRAINT <apribojimo_pav>
CHECK (<stulelio_pav> <> <stulpelio_pav>);

„Asymmetric“ apribojimas gali būti įgyvendinamas naudojant tris trigerius:

- Prieš trigerio vykdymo lygį, nustatyti visas paketo būsenas žinomas
- Ir po, eilutes lygio trigerį užfiksuoti visų eilučių pakitimus
- Ir po būsenos trigerį norimam procesui keisti.

Šis paketas yra naudojamas naujausiai įterptų/koreguotu eilučių rowids saugojimui pakete. Aprašome du masyvus: vienas išsaugos naujų eilučių id (rowids), kitas bus naudojamas šio masyvo išvalymui.

```
create or replace package state_pkg
as
  type ridArray is table of rowid index by binary_integer;
  newRows ridArray;
  empty   ridArray;
end;
```

Šis trigeris būtinai prieš įterpimo/koregavimo procesą išsiskirs vietą.

```
create or replace trigger reset
before insert or update on taskRelation
begin
    state_pkg.newRows := state_pkg.empty;
end;
```

Sekantis trigeris paprastai gauna paveiktos eilutės id ir išsaugo newRows masyve.

```
create or replace trigger saveRowID
after insert or update on taskRelation for each row
begin
    state_pkg.newRows( state_pkg.newRows.count+1 ) := :new.rowid;
end;
```

Trečias trigeris padaro kilpą kiekvienai įterptai ar pakeistai eilutei. Paimame rowed ir tikriname ar atvirkštinis variantas egzistuoja.

“Antisymmetric” apribojimas neleidžia atvirkščio ryšio taip kaip “asymmetric”, bet skirtingai nuo jo leidžia tam pačiam atvejui dalyvauti abejose apribojimo rolėse. Taigi realizavimas yra panašus į “asymmetric” apribojimo. Skiriasi tik trečio trigerio sąlyga, išsaukianti konfliktą.
“Intransitive” apribojimo trečiasis trigeris yra skirtingas. Jis turi išvesti klaidą jei įterptas įrašas pereinamas.

“Acyclic” apribojimo trečiasis trigeris yra skirtingas. Jis turi išvesti klaidą jei įterptas įrašas formuoja ciklą.

4.3.4 Vientisumo apribojimai objektų aibėms (Integrity constraints on sets of objects)

Išskirti penki vientisumo apribojimai: lygybės apribojimas nebūtiniems ryšiams ar atributams, poaibio apribojimas nebūtiniems atributams, išskyrimo apribojimas nebūtiniems atributams, išskyrimo apribojimas ryšiams. Jie pateikti 20 paveikslo lentelėje.

Apribojimas	**Naudojimas**	**Realizavimo galimybės**	**Pasirinktas realizavimo metodas**	**Atitikmuo DBVS**
Lygybės *apribojimas nebūtiniems ryšiams ar atributams (*Equality constraint on groups of optional relationships or attributes)	Stulpelių laukų užtikrintinumui nustatyti	Check funkcija, trigeriu	Check funkcija.	Nėra
***Poaibio apribojimas nebūtiniems atributams (*Subset constraint on optional attributes)**	Naudojamas nebūtiniems esybių atributams ir ryšiams. Tokiu atveju mes turime įvardinti visus leidžiamus variantus.	Check apribojomų funkcija arba trigeriu	Ir check funkcija ir trigeriu	Nėra
***Poaibio apribojimas nebūtiniems ryšiams (*Subset constraint on optional relationships)*:**	Siekiant uždrausti vienos lentelės asociacijas su kita.	Trigeriu	Trigeriu	Nėra
Išskyrimo apribojimas *nebūtiniems atributams* (Exclusion constraint on optional attributes)	Naudojamas norint lentelės laukams užtikrinti, kad vienas ir tas pats atributas gali turėti vieną arba kitą reikšmę, tačiau negali įgyti abieju kartu.	Naudojant CHECK apribojimų funkcijas	CHECK	Nėra
***Išskyrimo apribojimas ryšiams (*Exclusion constraint on optional attribute)**	Naudojamas kada turime ryšius m:n	Trigeriu	Trigeriu	Nėra

20 pav. Lentelė

Lygybės apribojimas nebūtiniems ryšiams ar atributams (**Equality constraint on groups of optional relationships or attributes)**

Vykdomas check apribojimų realizavimu lentelės stulpeliams.

```
ALTER TABLE <lenteles_pav>
ADD CONSTRAINT <apriboijimo_pav>
        CHECK (
        (<stulpelio_pav> IS NOT NULL AND < stulpelio_pav > IS NOT NULL ... < stulpelio_pav > IS NOT NULL)
        OR
(<stulpelio_pav > IS NULL AND < stulpelio_pav > IS NULL ... < stulpelio_pav > IS NULL));
```

Poaibio apribojimas nebūtiniems atributams **(Subset constraint on optional attributes)**

Naudojamas nebūtiniems esybių atributams ir ryšiams gali būti realizuojamas naudojant check apribojimus. Tokiu atvejų mes turime įvardinti visus leidžiamus variantus firsName ir lastName:

```
ALTER TABLE <lenteles_pav>
ADD CONSTRAINT subset
CHECK (
        (<lauko_pav> IS NOT NULL AND <lauko_pav1> IS NOT NULL)
        OR
        (<lauko_pav> IS NULL AND <lauko_pav1>  IS NOT NULL)
        OR
        (<lauko_pav> IS NULL AND <lauko_pav1>  IS NULL));
```

Kita galimybė poaibio apribojimo vykdymo, realizacija naudojant trigerį:

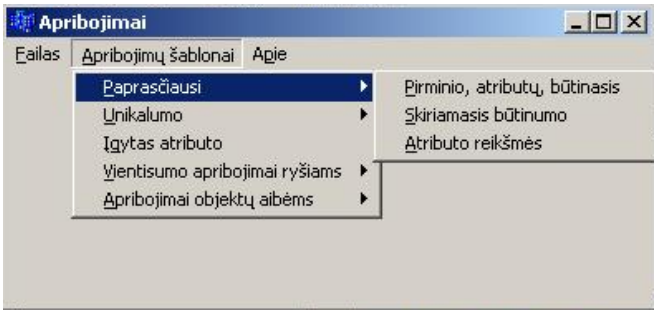
```
CREATE OR REPLACE TRIGGER <trigerio_pav>
  BEFORE INSERT OR UPDATE OF <lauko_pav>, <lauko_pav1> on <lenteles_pav>
  for each row
DECLARE
  Invalid_record EXCEPTION;
BEGIN
  IF :new.<lauko_pav> IS NOT NULL AND :new.<lauko_pav1>  IS NULL THEN
    RAISE Invalid_record;

  END IF;

  EXCEPTION
     WHEN Invalid_record THEN
       RAISE_APPLICATION_ERROR (
         num=> -20107,
         msg=> 'Pranesimas!');
END <trigerio_pav>;
```

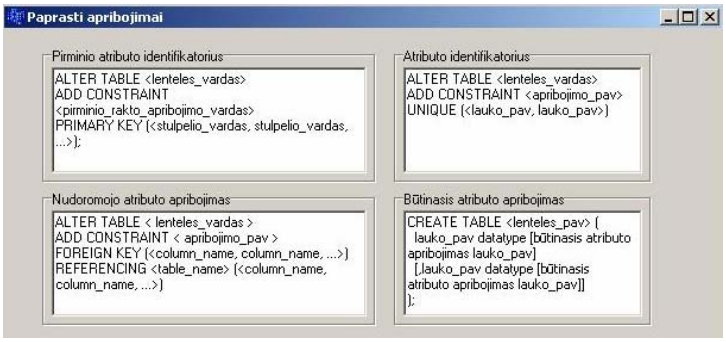
4.4 Šablonų sistema

Pagal sukurtą ir išanalizuotą apribojimų metodiką, sukurta nedidelė dialogo tipo sistema pateikta 21 paveiksle , padėsianti vartotojui realizuoti vienokį ar kitokį apribojimo tipą.

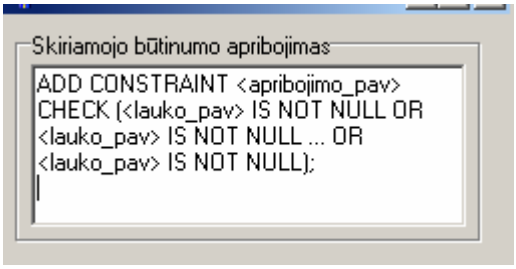


21 pav.

Per meniu pasirinkus paprasčiausių apribojimų realizavimą sistema pateikia langą, sų apribojimų šablonais 22 paveiksle.

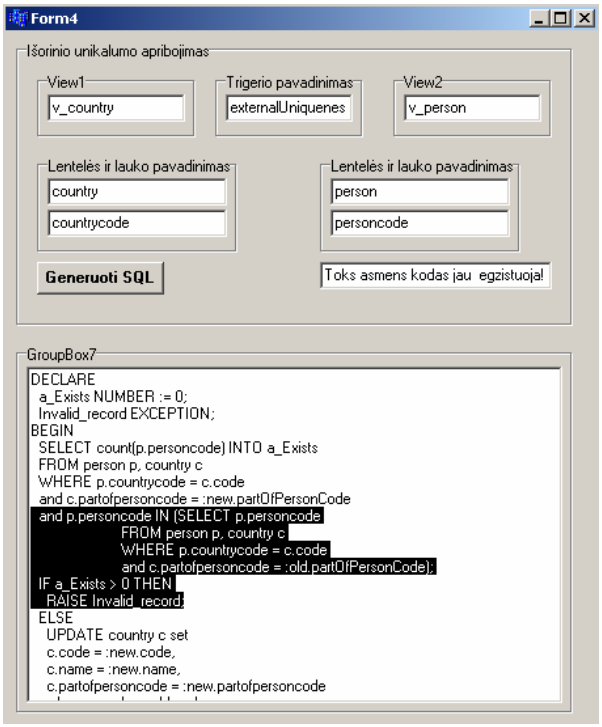


22 pav.

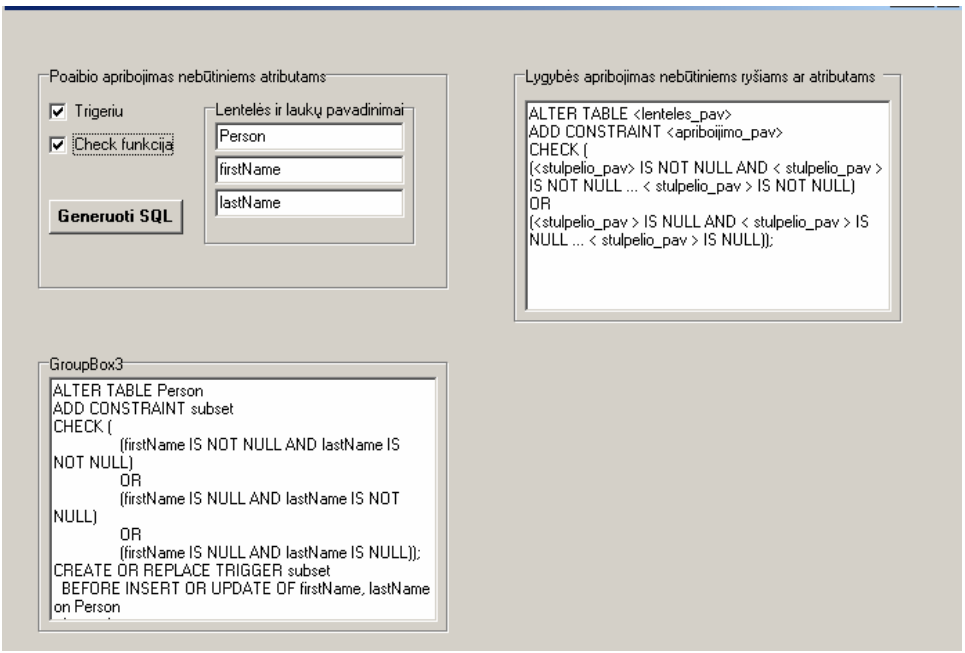


23 pav.

Meniu pasirinkus išorinio unikalumo apribojimo šablono kūrimą, reikia atitinkamai užpildyti laukus, ir paspaudus mygtuką sistema sugeneruoja PL/SQL kodą skirta šiam unikalumo apribojimui realizuoti. Toliau pateikta keletas apribojimo šablonų realizavimo dialogo langų 24 ir 25 paveikslas.



24 pav.



25 pav.

5. Apribojimų realizavimas ir testavimas

5.1 Apribojimų realizavimo pavyzdžiai ir testavimas

Pirminis atributo identifikatorius: (arba grupės atributų). **(Primary identifier on attribute)**

Naudojimas:

Pirminis atributo identifikatorius yra naudojamas lentelės lauko raktui nusakyti.

Kuriant duomenų bazės lentelę, DBVS tą gali padaryti pagal nutylėjimą, jei aprašytą, lauko tipą įvardinsime kaip PRIMARY KEY.

```
CREATE TABLE Country1 (
code varchar(20) PRIMARY KEY,
name varchar(20),
partOfPersonCode varchar(20))
```

Pirminio identifikatoriaus naudojimas, lentelės stulpeliams ORACLE 10g sintaksėje:

```
ALTER TABLE <table_name>
ADD CONSTRAINT <primary_key_constraint_name>
PRIMARY KEY (<column_name, column_name, ...>);

ALTER TABLE <Lenteles_vardas>
ADD CONSTRAINT <pirminio_rakto_apribojimo_vardas>
PRIMARY KEY (<stulpelio_vardas, stulpelio_vardas, ...>);
```

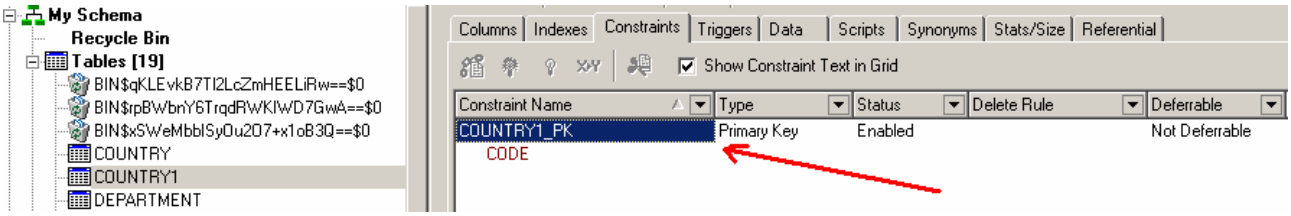
Atliekame šio tipo apribojimo, sukūrimą, bei testavimą:

Šiam tikslui buvo sukurta lentele pavadinimu “Country”:

```
CREATE TABLE Country1 (
code varchar(20),
name varchar(20),
partOfPersonCode varchar(20))
```

Iš paveikslo matome, kad lentelė neturi jokio apribojimo. Dabar pasinaudojant pirminio atributo identifikavimo šablonu sukursime šį apribojimą:

```
ALTER TABLE Country1
ADD CONSTRAINT Country1_PK
PRIMARY KEY (code);
```



26 pav.

Kaip pastebime iš šio paveikslo, kad dabar lentelė turi pirminio atributo identifikavimo apribojimą.

Atributo identifikatorius: (arba grupės atributų). **(Identifier constraint on attribute)**

Naudojimas:

Atributo identifikatorius yra naudojamas lentelės lauko unikalumui (unique) nusakyti.

Kuriant duomenų bazės lentelę, DBVS tą gali padaryti pagal nutylėjimą, jei aprašytą, lauko tipą įvardinsime kaip UNIQUE.

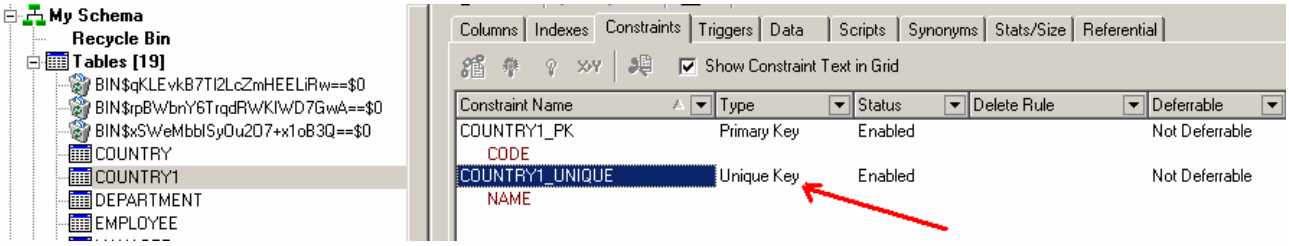
```
CREATE TABLE Country1 (
code varchar(20),
name varchar(20) UNIQUE,
partOfPersonCode varchar(20))
```

```
ALTER TABLE <table_name>
ADD CONSTRAINT <constraint_name>
UNIQUE (<column_name, column_name, ...>);
```

```
ALTER TABLE <lenteles_vardas>
ADD CONSTRAINT <apribojimo_pav
UNIQUE (<lauko_pav, lauko_pav>)
```

Pavyzdys:

```
ALTER TABLE Country1
ADD CONSTRAINT Country1_UNIQUE
UNIQUE (name);
```



27 pav.

Kaip ir prieš tai atliktoje procedūroje su pirminiu atributu, tai dabar matome, kad lentelės laukas „name“ turi unikalaus identifikavimo apribojimą.

Nurodomasis atributo apribojimas: **(Referential constraint on attribute)**

Naudojimas:

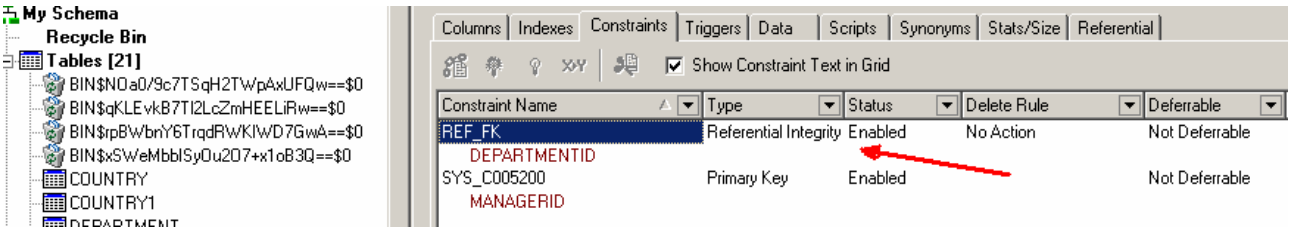
Šis apribojimas DBVS yra naudojamas norint susieti lenteles tarpusavio ryšiu. T.y. šio proceso metu yra susiejamas lentelės laukas, turintis pirminio atributo apribojimą, ir lentelės laukas, kuriame nusakytas nurodomojo atributo apribojimas.

ALTER TABLE <table_name>
ADD CONSTRAINT <constraint_name>
FOREIGN KEY (<column_name, column_name, ...>)
REFERENCING <table_name> (<column_name, column_name, ...>)

Sukurtai lentelei „Manager1“ ir „Department“ sukursime ryšį tarp lentelių, tuo pačiu sukuriamas ir Nurodomasis atributo apribojimas.

CREATE TABLE Manager1(
managerID integer PRIMARY KEY,
departmentID integer)

ALTER TABLE Manager1
ADD CONSTRAINT REF_FK
FOREIGN KEY (departmentID)
REFERENCING Department (departmentID)



28 pav.

Kaip ir prieš tai atliktoje procedūroje su tai dabar matome, kad DBVS egzistuoja sukurtas nurodomojo tipo apribojimas.

Būtinasis atributo apribojimas **(Mandatory constraint on attribute)**

Naudojamas:

Lentelės kūrimo procese nustatomas būtinas apribojimas laukui, kuris privalo turėti kažkokią reikšmę .

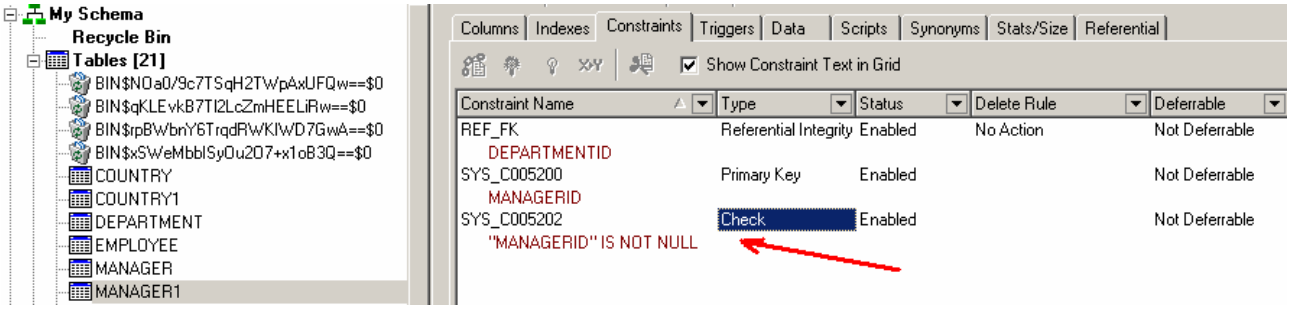
CREATE TABLE <lenteles_pav> (
  lauko_pav datatype [būtinasis atributo apribojimas lauko_pav]
  [,lauko_pav datatype [būtinasis atributo apribojimas lauko_pav]]
);

Pridedant būtinumo apribojimą esamai lentelei:

ALTER TABLE < lenteles_pav > MODIFY < lauko_pav > NOT NULL;

Apribojimo pritaikymas duomenų bazės lentelei „Manager“:

ALTER TABLE Manager1 MODIFY ManagerID NOT NULL;



29 pav.

Nenurodžius apribojimo vardo, duomenų bazės valdymo sistema, vardą parenka pagal nutylėjimą. Šiuo atveju SYS_C005202. šiam apribojimui realizuoti DBVS naudoja funkcija CHECK

Kaip matome iš pateikto paveikslo, lentelė šiuo metu turi realizuotus tris apribojimų tipus.

Skiriamasis būtinumo apribojimas: **(Disjunctive mandatory constraint)**

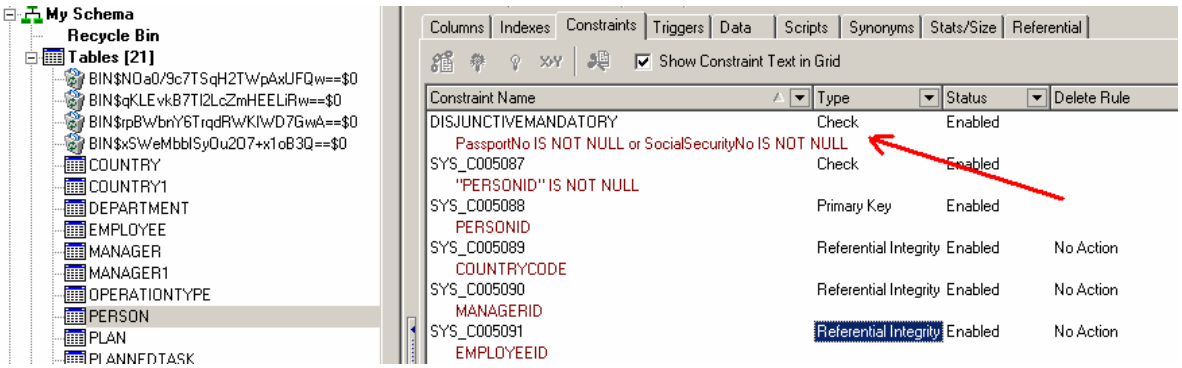
Naudojimas:

Šis skiriamasis būtinumo apribojimas naudojamas siekiant patikrinti vienokio ar kitokio lauko reikšmę:

ALTER TABLE <lenteles_pav>
ADD CONSTRAINT <apribojimo_pav>
CHECK (<lauko_pav> IS NOT NULL OR <lauko_pav> IS NOT NULL ... OR <lauko_pav> IS NOT NULL);

Kaip pavyzdys šio apribojimo lentelės „Person“ du laisvai pasirenkami laukai PassportNo ir SocialSecurityNo:

ALTER TABLE Person
ADD CONSTRAINT disjunctiveMandatory
CHECK (PassportNo IS NOT NULL or SocialSecurityNo IS NOT NULL);



30 pav.

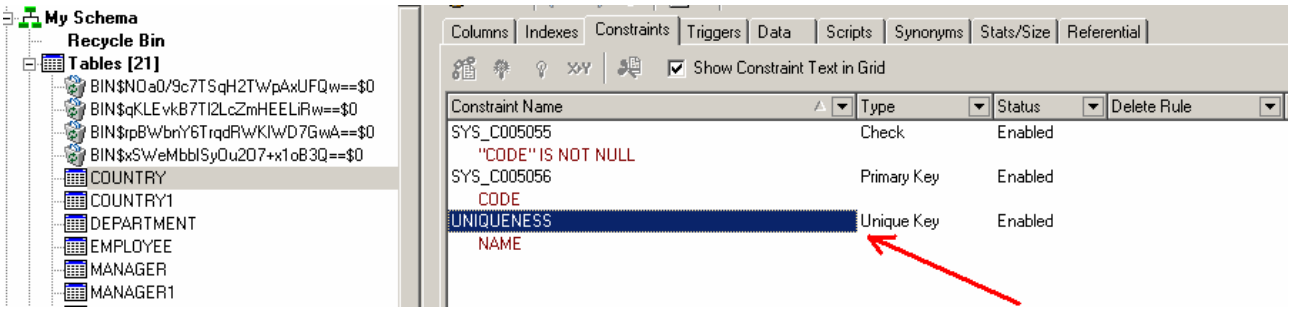
Vidinis unikalumo apribojimas (*Internal uniqueness constraint)*:

Naudojimas:

Vidinis atributo apribojimas yra naudojamas lentelės lauko unikalumui nusakyti.

Atliekame su duomenų bazės lentele “Country”

```
ALTER TABLE Country
ADD CONSTRAINT uniqueness
        UNIQUE (name);
```



31 pav.

***Išorinis unikalumo apribojimas:* (External uniqueness constraint)**

Naudojimas:

Kai reikia identifikuoti keliose lentelėse susietus egzistuojančius atributus.

Atliksime išorinio unikalumo apribojimo realizaciją. Atliekant

Išorinio unikalumo apribojimo realizacijai atlikti naudojami vaizdai (view) lentelėse, kuriose šis apribojimas bus taikomas.

Šis apribojimas bus taikomas Country ir Person lentelėse. Sukuriame Country lentelės view pavadinimu v_country:

```
CREATE OR REPLACE VIEW v_country AS
SELECT "code","name","partOfPersonCode" FROM Country;
```

CODE	NAME	PARTOFPERSONCODE
EU0001	Vengrija	A01
EU0002	Austrija	A02
EU0003	Lietuva	A03
EU0006	Rusija	A06
EU0004	Ispanija	A04

32 pav.

Taip pat analogiškai Person lentelės view pavadinimu v_person:

```
CREATE OR REPLACE VIEW v_person AS
SELECT "personID","firstName","lastName","countryCode","personCode",
"passportNo","socialSecurityNo","sex","birthday","age","managerID",
        "employeeID" FROM Person;
```

PERSONID	FIRSTN...	LASTNAME	COUN...	PERSON...	PASSPO...	SOCIALSEC...	S...	BIRTHD...
10001	Elvis	Kadraitis	EU0003	3820130521	LK311125	asd4654654	M	1981.10.09
10002	Martynas	Abromavicius	EU0003	3820526254	LK60014	klasd545454	M	1982.05.26
111111	Edvinas	Eidukevicius	EU0002	3824545425	LK50099	xxx6909d4	M	1982.01.30
777777	Jonas	Salvis	EU0003	1111111111	LK14522		M	1979.07.10

33 pav.

Šių sukurtų vaizdų paskirtis būtų lentelėje Country uždrausti koreguoti Country lentelės atributą PartOfPersonCode jei koregavimo operacija sukeltų išorinio apribojimo pažeidimus. Įterpimo operacijos nėra stebimos, kadangi jos nepažeis išorinio unikalumo apribojimų.

Tolesniems apribojimo realizacijos darbams naudosime specialiai vaizdams (view) naudojamus trigerius INSTEAD OF.

```
CREATE OR REPLACE TRIGGER externalUniquenessC
  INSTEAD OF UPDATE ON V_COUNTRY
  FOR EACH ROW
DECLARE
  a_Exists NUMBER := 0;
  Invalid_record EXCEPTION;
BEGIN
  SELECT count(p.personcode) INTO a_Exists
  FROM person p, country c
  WHERE p.countrycode = c.code
  and c.partofpersoncode = :new.partOfPersonCode
  and p.personcode IN (SELECT p.personcode
                       FROM person p, country c
                       WHERE p.countrycode = c.code
                       and c.partofpersoncode = :old.partOfPersonCode);
  IF a_Exists > 0 THEN
    RAISE Invalid_record;
  ELSE
    UPDATE country c set
```

```
    c.code = :new.code,
    c.name = :new.name,
    c.partofpersoncode = :new.partofpersoncode
    where c.code = :old.code;
  END IF;
  EXCEPTION
     WHEN Invalid_record THEN
       RAISE_APPLICATION_ERROR (
         num=> -20107,
         msg=> 'Toks asmens kodas jau egzistuoja');
END externalUniquenessC;
```

Įterpimo ir koregavimo operacijos lentelėje Person gali pažeisti išorinio unikalumo apribojimą. Tam reikalingas trigeris koregavimo ir įterpimo operacijai lentelei Person:

```
CREATE OR REPLACE TRIGGER externalUniquenessV
  INSTEAD OF INSERT OR UPDATE ON v_PERSON
  FOR EACH ROW
DECLARE
  a_Exists NUMBER := 0;
  Invalid_record EXCEPTION;
BEGIN
   SELECT count(temp.rowid) INTO a_Exists
   FROM (SELECT c.partofpersoncode, p.personcode, p.personID
         FROM person p, country c
         WHERE p.countrycode = c.code) temp, country cc
   WHERE temp.partOfPersonCode = cc.partofpersoncode
   AND   cc.code = :new.countryCode
   AND   temp.personCode = :new.personCode
   AND   temp.personID <> :old.personID;
   IF a_Exists > 0 THEN
     RAISE Invalid_record;
   ELSIF updating THEN
     update person p set
     p.personID = :new.personID,
     p.firstName = :new.firstName,
     p.lastName = :new.lastName,
     p.countryCode = :new.countryCode,
     p.personCode = :new.personCode,
     p.passportNo = :new.passportNo,
     p.socialSecurityNo = :new.socialSecurityNo,
     p.sex = :new.sex,
     p.birthday = :new.birthday,
     p.age = :new.age,
     p.managerID = :new.managerID,
     p.employeeID = :new.employeeID
     where p.personID = :old.personID;
   ELSIF inserting THEN
     INSERT INTO person(personID, firstName, lastName, countryCode, personCode,
     passportNo, socialSecurityNo, sex, birthday, age, managerID, employeeID)
     VALUES(:new.personID, :new.firstName, :new.lastName, :new.countryCode,
:new.personCode, :new.passportNo, :new.socialSecurityNo, :new.sex,
:new.birthday, :new.age, :new.managerID, :new.employeeID);
   END IF;
  EXCEPTION
    WHEN Invalid_record THEN
```

```
    RAISE_APPLICATION_ERROR (
      num=> -20107,
      msg=> 'Toks asmens kodas jau egzistuoja');
END externalUniquenessV;
```

Apribojimo testavimas:

Tikslas: Įsitikinti ar realizuotas apribojimas veikia užtikrintai.

Lentelės Country ir Person susietos tarpusavio ryšiu. Lentelėje Person gali būti daugybės asmenų turinčių asmens kodą. Gali pasitaikyti atvejų kai žmonės iš skirtingų šalių turės vienodą asmens kodą. Tam, kad unikaliai identifikuoti šiuos žmones naudojamas šis apribojimas.

Etapai:

1. Patikrinti ar lentelėje Country redaguojant atributą PartOfPersonCode apribojimas neleidžia to atlikti esant galimybei pažeisti unikalumą t.y. asmens kodas negali dubliuotis su iš pačios šalies esančio žmogaus asmens kodu.
2. Patikrinti ar lentelėje Person atliekant naujo asmens įvedimą ar redagavimą apribojimas neleidžia to atlikti esant galimybei pažeisti unikalumą t.y. įvedamas ar redaguojamas asmens kodas negali jau egzistuoti su iš pačios šalies esančio žmogaus asmens kodu.

Lentelės Country vaizdo (view) v_country duomenys:

CODE	NAME	PARTOFPERSONCODE
EU0001	Vengrija	A01
EU0002	Austrija	A02
EU0003	Lietuva	A03
EU0006	Rusija	A06
EU0004	Ispanija	A04

34 pav.

Lentelės Person vaizdo (view) v_person duomenys:

PERSO...	FIRSTNAME	LASTNAME	COUNTRYCODE	PERSONCODE	PASSPORTNO
10002	Martynas	Abromavicius	EU0003	38205262541	LK60014
100021	Elvis	Kadraitis	EU0003	38201305214	LK311125
111111	Edvinas	Eidukevicius	EU0002	38245454255	jx50099
777777	Jonas	Salvis	EU0003	11111111111	LK14522
4444444	Romas	Danilevicius	EU0001	38201305214	DSN45112

35 pav.

Apribojimas neturi leisti atlikti redagavimo operacijos jeigu yra norima pakeisti PartOfPersonalCode reikšmę A03 į A01, kadangi lentelėje Person egzistuoja žmogus turintis toki pat asmens kodą, tačiau iš kitos šalies. Operacijos metu mes norime atlikti koregavimą t.y. visus esančius įrašus susijusius su Lietuva priskirti Vengrijai, kai tuo metu asmenys ID = 100021 Elvis Kadraitis personcode = 38201305214 ir ID = 4444444 Romas Danilevicius personcode = 38201305214 turi vienodą asmens kodą, apribojimas turėtų neleisti to padaryti, kitu atveju jei sutampančių asmens kodų nebūtų, apribojimas leistų vykdyti pakitimus.

Lentelė	Atributas			
Country	PartOfPersonCode			
Pradinė reikšmė	Galimos reikšmės	Gautas rezultatas	Laukiamas rezultatas	Atitinka
A03	A01	ORA-20107: Toks asmens kodas jau egzistuoja	Pranešimas ORA-20107: Toks asmens kodas jau egzistuoja	√
	A02	A02	Lauko reikšmės pakeitimas į A02	√
	A05	A05	Lauko reikšmės pakeitimas į A05	√

36 pav.

Lentelė	Atributas			
Country	PartOfPersonCode			
Pradinė reikšmė	Galimos reikšmės	Gautas rezultatas	Laukiamas rezultatas	Atitinka
A01	A03	ORA-20107: Toks asmens kodas jau egzistuoja	Pranešimas ORA-20107: Toks asmens kodas jau egzistuoja	√
	A02	A02	Lauko reikšmės pakeitimas į A02	√
	A05	A05	Lauko reikšmės pakeitimas į A05	√

37 pav.

Apribojimas neturi leisti atlikti redagavimo operacijos jeigu yra norima pakeisti CountryCode ar personCode reikšmes, kadangi lentelėje Person egzistuoja žmogus turintis toki pat asmens kodą, tačiau iš kitos šalies. Operacijos metu mes norime atlikti koregavimą t.y. pakeisti žmogaus asmens kodą į jau egzistuojantį iš tos pačios šalies.

Lentelė	Atributas			
Person	CountryCode			
Pradinė reikšmė	Galimos reikšmės	Gautas rezultatas	Laukiamas rezultatas	Atitinka
EU0003	EU0001	ORA-20107: Toks asmens kodas jau egzistuoja	Pranešimas ORA-20107: Toks asmens kodas jau egzistuoja	√
	EU0002	EU0002	Lauko reikšmės pakeitimas į EU0002	√

	EU0005	EU0005	Lauko reikšmės pakeitimas į EU0005	√

38 pav.

Lentelė	Atributas			
Person	CountryCode			
Pradinė reikšmė	Galimos reikšmės	Gautas rezultatas	Laukiamas rezultatas	Atitinka
EU0001	EU0003	ORA-20107: Toks asmens kodas jau egzistuoja	Pranešimas ORA-20107: Toks asmens kodas jau egzistuoja	√
	EU0002	EU0002	Lauko reikšmės pakeitimas į EU0002	√
	EU0005	EU0005	Lauko reikšmės pakeitimas į EU0005	√

39 pav.

Lentelė	Atributas			
Person	CountryCode			
Pradinė reikšmė	Galimos reikšmės	Gautas rezultatas	Laukiamas rezultatas	Atitinka
EU0002	EU0001	EU0001	Lauko reikšmės pakeitimas į EU0003	√
	EU0003	EU0003	Lauko reikšmės pakeitimas į EU0003	√
	EU0005	EU0005	Lauko reikšmės pakeitimas į EU0005	√

40 pav.

Atributo reikšmės apribojimas **(Value constraint on attribute)**

Naudojimas: Uždraudžiant atributo reikšmes baigiamojoje reikšmių nurodymo stadijoje, įvertinant pradines ir galutines reikšmes (intervalą).

Šis apribojimas realizuojamas naudojant CHECK funkciją.

Pateiktas reikšmės apribojimas, kada leidžiamos reikšmės yra įvardintos, panaudojant check apribojimą:

ALTER TABLE <lenteles_pav>
ADD CONSTRAINT <apribojimo_pav>
CHECK (<stulpelio_pav > = <reiksme> OR < stulpelio_pav > = <reiksme > ... OR < stulpelio_pav > = <reiksme >);

Pateiktas reikšmės apribojimas, kada leidžiamos reikšmės yra apibrėžtos nurodant intervalą, panaudojant check apribojimą:

ALTER TABLE < lenteles_pav >
ADD CONSTRAINT < apribojimo_pav >
CHECK (<stulpelio_pav > BETWEEN <reiksme> AND <reiskme> ... OR < stulpelio_pav > BETWEEN <reiksme> AND <reiksme>);

Pateiktas reikšmės apribojimas, kada leidžiamos reikšmės yra apibrėžtos nurodant intervalą ar įvardintos, panaudojant check apribojimą:

ALTER TABLE < lenteles_pav >
ADD CONSTRAINT < apribojimo_pav >
CHECK (<stulpelio_pav > = <reiksme> OR < stulpelio_pav > = <reiksme> ... OR < stulpelio_pav > BETWEEN <reiskme> AND <reiskme> ... OR < stulpelio_pav > BETWEEN <reiksme> AND <reiksme>));

Pavyzdys mūsų duomenų bazėje:

ALTER TABLE PERSON
ADD CONSTRAINT value
CHECK (sex='M' or sex='F');

Atliekame apribojimo vykdymo testavimą. Bandysime įvesti, bei koreguoti lauko reikšmes. Leistinos reikšmės yra ‘M’ ir ‘V’.

Lentelė	Atributas			
Person	Sex			
Pradinė reikšmė	Galimos reikšmės	Gautas rezultatas	Laukiamas rezultatas	Atitinka
M, V	254	ORA-02290: check constraint (LABAS.VALUE) violated	Pranešimas Sisteminis, kad pažeistas apribojimas	√
	M	M	Lauko reikšmės pakeitimas į M	√
	vyras	ORA-02290: check constraint (LABAS.VALUE) violated	Pranešimas Sisteminis, kad pažeistas apribojimas	√
	V	V	Lauko reikšmės pakeitimas į V	√

41 pav.

Išvestinio atributo apribojimas **(Derived attribute constraint)**

Naudojimas: Jei gautas atributas yra skaičiuojamas iš atributų, kurie priklauso tai pačiai lentelei, tokiu atveju gauto atributo skaičiavimo taisyklė gali būti vykdoma trigeryje. Prieš INSERT ar UPDATE kiekvieno lentelės įrašo bus paruošiama nauja gauto atributo reikšmė.

Kaip pavyzdį galime paimti trigerį skirtą žmogaus amžiaus skaičiavimui:

```
CREATE OR REPLACE TRIGGER derivedAttribute
  BEFORE INSERT OR UPDATE OF Birthday ON Person
  for each row
BEGIN
  :new.age := trunc((sysdate - :new.birthDay)/365,0);
END derivedAttribute;
```

Lentelė	Atributas			
Person	Birthday			
Pradinė reikšmė	Galimos reikšmės	Buves/Gautas „Age“ lauke	Laukiamas rezultatas	Atitinka
1982.05.18	1982.05.19	23/23	23/23	√
1989.01.10	1989.01.01	16/17	16/17	√
1989.01.01	1989.10.01	17/16	17/16	√

42 pav.

Jei gaunamas atributas yra skaičiuojamas iš atributų, priklausančių skirtingoms lentelėms, tokiu atveju gaunamo atributo reikšmė turėtų būti perskaičiuojama, kada mes įterpiame, koreguojame ar ištriname įrašą.

```
CREATE OR REPLACE TRIGGER derivedAttribute
  AFTER INSERT OR UPDATE of uzsakymoNr, kiekis, kaina OR DELETE ON UzsakymoEilutes
  for each row
BEGIN
  IF INSERTING THEN

     update uzsakymas u set
     u.uzsakymosuma = nvl(u.uzsakymosuma,0) + :new.kiekis*:new.kaina
     where u.uzsakymonr = :new.uzsakymoNr;

  ELSIF UPDATING THEN

    IF :old.uzsakymoNr = :new.uzsakymoNr THEN
       update uzsakymas u set
       u.uzsakymosuma = u.uzsakymosuma + :new.kiekis*:new.kaina - :old.kiekis*:old.kaina
       where u.uzsakymonr = :new.uzsakymoNr;
    ELSE
       update uzsakymas u set
       u.uzsakymosuma = u.uzsakymosuma - :old.kiekis*:old.kaina
       where u.uzsakymonr = :old.uzsakymoNr;

       update uzsakymas u set
       u.uzsakymosuma = u.uzsakymosuma + :new.kiekis*:new.kaina
       where u.uzsakymonr = :new.uzsakymoNr;
    END IF;

  ELSIF DELETING THEN

     update uzsakymas u set
     u.uzsakymosuma = u.uzsakymosuma - :old.kiekis*:old.kaina
     where u.uzsakymonr = :old.uzsakymoNr;

  END IF;

END derivedAttribute;
```

Vientisumo apribojimai ryšiams (Integrity constraints on relationship or relationships)

Daugialypiškumo apribojimas (Multiplicity constraints):

Daugialypiškumo apribojimas, kuris uždraudžia lentelės Module1 įrašų kiekį, susijusių su lentelės Student vienu įrašu esant didesnėm ar mažesnėm ribom, gali būti įgyvendinamas naudojant trigeriu.

```
CREATE OR REPLACE TRIGGER cardinality
  AFTER INSERT OR UPDATE ON Module1
DECLARE
  a_Count NUMBER := 0;
  Invalid_record EXCEPTION;
BEGIN
  Select SID,(count(SID))
  from Module1 B
  group by B.SID
  having count(b.sid)>2 and count(b.sid)<5
  If (a_Count > 0) Then
    RAISE Invalid_record;
  END IF;
  EXCEPTION
     WHEN Invalid_record THEN
       RAISE_APPLICATION_ERROR (
         num=> -20107,
         msg=> 'Daugialypiskumo apribojimo pazeidimas');
     WHEN NO_DATA_FOUND THEN NULL;
END cardinality;
```

Apribojimas neturi leisti atlikti redagavimo, įterpimo operacijos jeigu studento ID yra susijęs su dviem ar mažiau modulių skaičiumi arba daugiau kaip 4 moduliais. T.y. kiekvienas studentas Module lentelėje privalės turėti nuo 3 iki 4 studijų modulių imtinai.

Lentelė	Atributas			
Module	SID			
SID reikšmė	Įvedamos reikšmės MID	Gautas rezultatas	Laukiamas rezultatas	Atitinka
01	M01, M02	ORA-20107: Daugialypiskumo apribojimo pazeidimas	Pranešimas kad pažeistas daugialypiškumo apribojimas	√
	M01, M02, Nk11	Sistema leido įvesti modulius	Įterpimo įvykdymas	√
	M02, M03, M04, ZXC1	Sistema leido įvesti modulius	Įterpimo įvykdymas	√
	M02, M03, M04, Z13, Z212	ORA-20107: Daugialypiskumo apribojimo pazeidimas	Pranešimas kad pažeistas daugialypiškumo apribojimas	√

43 pav.

Daugialypiškumo apribojimas, kuris riboja lentelės Module įrašų kiekį, susijusių su lentelės Student vienu įrašu, esant didesnei ribai, gali būti įgyvendinamas naudojant vaizdams (view) skirtus trigerius.

```
CREATE OR REPLACE TRIGGER cardinalityV
  INSTEAD OF INSERT OR UPDATE on V_Module
   ore ach row
DECLARE
  a_Count NUMBER := 0;
  Invalid_record EXCEPTION;
BEGIN
  select count(b.sid) into a_Count
  from Module B
  where b.sid = :new.sid;
  If :new.Sid <> :old.sid and a_Count >= 2 Then
    RAISE Invalid_record;
  ELSE
     IF INSERTING THEN
      INSERT INTO Module(SID, MID)
      VALUES (:new.sid, :new.mid);
    ELSIF UPDATING THEN
      UPDATE Module SET
      module.sid = :new.sid
      WHERE module.MID = :new.mid;
    END IF;
  END IF;
  EXCEPTION
     WHEN Invalid_record THEN
       RAISE_APPLICATION_ERROR (
         num=> -20107,
         msg=> ‚Daugialypiskumo apribojimo pazeidimas‘);
END cardinalityV;
```

Apibendrinimo/specifikavimo ryšių apribojimai (Constraints on generalization/specialization relationship).

Yra trys pagrindinės strategijos, kurios yra naudojamos įgyvendinant apibendrinimo/specifikavimo ryšius:

- Visos klasės hierarchiškai prijungtos prie vienos lentelės. Lentelė turi lauką, kuris reikalingas kaip lauko diskriminatorius, tai yra laukas, kurio reikšmes identifikuoja specifinė subklasė, kuriai eilutei priklauso.
- Kiekviena subklasė yra priskirta skirtingai lentelei. Kiekviena lentelė gauna lauką visiems atributams super tipo su originaliu pasirenkamumu.

- ❖ Hierarchijų super klasė, pateikiama viena lentele. Kiekviena subklasė yra pateikiama skirtinga lentele, kuri saugo atitinkamai svarbius subklasės laukus, taip kaip laukai yra pateikiami pirminio rakto pavidalu.

Nepaisant visų šitų variantų privalumų ir trūkumų, reikia pasirinkti, kokia strategija bus naudojama. Bet nei viena iš jų nėra gera, jei mes naudosime apribojimus apibendrinimo/specifikavimo ryšiams.

CHECK apribojimo *išskirstymo* apribojimui dviems papildomiems stulpeliams sukurtiems pagrindinėje lentelėje:

```
ALTER TABLE <lenteles_pav>
ADD CONSTRAINT <apribojimo_pav>
        CHECK ((<stulpelio_pav> IS NOT NULL AND < stulpelio_pav > IS NULL) OR
        (<stulpelio_pav > IS NULL AND < stulpelio_pav > IS NOT NULL) OR
        (<stulpelio_pav > IS NULL AND < stulpelio_pav > IS NULL));
```

CHECK apribojimo *pilnam* apribojimui dviems papildomiems stulpeliams sukurtiems pagrindinėje lentelėje:

```
ALTER TABLE < lenteles_pav >
ADD CONSTRAINT < apribojimo_pav >
        CHECK (<stulpelio_pav > IS NOT NULL OR < stulpelio_pav > IS NOT NULL);
```

CHECK apribojimo *pilnam* ir *išskirstymo* apribojimui dviems papildomiems stulpeliams sukurtiems pagrindinėje lentelėje:

```
ALTER TABLE < lenteles_pav >
ADD CONSTRAINT < apribojimo_pav >
        CHECK ((<stulpelio_pav > IS NOT NULL AND < stulpelio_pav > IS NULL) OR
        (<stulpelio_pav > IS NULL AND < stulpelio_pav > IS NOT NULL));
```

Visi trys tipai realizuoti sistemoje:

```
ALTER TABLE Person
ADD CONSTRAINT disjoint
CHECK ((employeeID IS NOT NULL AND managerID IS NULL) OR
                (employeeID IS NULL AND managerID IS NOT NULL) OR
                (employeeID IS NULL AND managerID IS NULL));
```

Lentelė	Atributai
Person	employeeID, managerID

employeeID	managerID	Gautas rezultatas	Laukiamas rezultatas	Atitinka

2	null	Sistema leido įvesti duomenis	Pranešimas Sisteminis, kad pažeistas apribojimas	√
null	null	Sistema leido įvesti duomenis	Įterpimo įvykdymas	√
null	4	Sistema leido įvesti duomenis	Įterpimo įvykdymas	√
2	8	ORA-02290: check constraint (LABAS.COMPLETE) violated	Pranešimas Sisteminis, kad pažeistas apribojimas	√

44 pav.

```
ALTER TABLE Person
ADD CONSTRAINT complete
        CHECK (employeeID IS NOT NULL OR managerID IS NOT NULL);
```

Lentelė	Atributai
Person	employeeID, managerID

employeeID	managerID	Gautas rezultatas	Laukiamas rezultatas	Atitinka
2	null	Sistema leido įvesti duomenis	Pranešimas Sisteminis, kad pažeistas apribojimas	√
null	null	ORA-02290: check constraint (LABAS.COMPLETE) violated	Įterpimo įvykdymas	√
null	4	Sistema leido įvesti duomenis	Įterpimo įvykdymas	√
2	8	Sistema leido įvesti duomenis	Pranešimas Sisteminis, kad pažeistas apribojimas	√

45 pav.

```
ALTER TABLE Person
ADD CONSTRAINT disjointComplete
CHECK ((employeeID IS NOT NULL AND managerID IS NULL) OR
        (employeeID IS NULL AND managerID IS NOT NULL));
```

Lentelė	Atributai
Person	employeeID, managerID

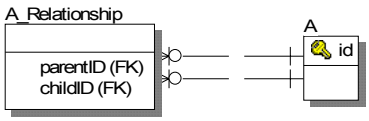
employeeID	managerID	Gautas rezultatas	Laukiamas rezultatas	Atitinka
2	null	Sistema leido įvesti duomenis	Pranešimas Sisteminis, kad pažeistas apribojimas	√

null	null	ORA-02290: check constraint (LABAS.COMPLETE) violated	Įterpimo įvykdymas	√
null	4	Sistema leido įvesti duomenis	Įterpimo įvykdymas	√
2	8	ORA-02290: check constraint (LABAS.COMPLETE) violated	Pranešimas Sisteminis, kad pažeistas apribojimas	√

46 pav.

Refleksyvių ryšių apribojimai: **(Constraints on reflexive associations)**

Yra šeši apribojimų tipai refleksyviems ryšiams.
Irreflexive, Asymmetric, Acyclic, Intransitive, Symmetric, Antisymmetric.



Paveikslėlyje matyti refleksinio ryšio pavyzdys. Reliaciniame modelyje galima sujungti skirtingais būdais. Panaudosime pagrindinę lentelę A ir viena papildomą lentelę A_Relationship ryšių saugojimui tarp lentelės A įrašų. Apribojimų vykdyme reikės naudoti check apribojimus arba trigerius, kad užtikrinti visas įmanomas atributų kombinacijas parentID ir childID.

“Irreflexive” apribojimas gali būti įgyvendinamas naudojant check apribojimus:

```
ALTER TABLE <lenteles_pav>
ADD CONSTRAINT <apribojimo_pav>
        CHECK (<stulelio_pav> <> <stulpelio_pav>);
```

„Asymmetric“ apribojimas gali būti įgyvendinamas naudojant tris trigerius:

- Prieš trigerio vykdymo lygį, nustatyti visas paketo būsenas žinomas
- Ir po, eilutes lygio trigerį užfiksuoti visų eilučių pakitimus
- Ir po būsenos trigerį norimam procesui keisti.

Šis paketas yra naudojamas naujausiai įterptų/koreguotu eilučių rowids saugojimui pakete. Aprašome du masyvus: vienas išsaugos naujų eilučių id (rowids), kitas bus naudojamas šio masyvo išvalymui.

```
Create or replace package state_pkg
as
   type ridArray is table of rowid index by binary_integer;
   newRows ridArray;
   empty   ridArray;
end;
```

Šis trigeris būtinai prieš įterpimo/koregavimo procesą išsiskirs vietą.

```
Create or replace trigger reset
before insert or update on taskRelation
begin
    state_pkg.newRows := state_pkg.empty;
end;
```

Sekantis trigeris paprastai gauna paveiktos eilutės id ir išsaugo newRows masyve.

```
Create or replace trigger saveRowID
after insert or update on taskRelation □ore ach row
begin
    state_pkg.newRows( state_pkg.newRows.count+1 ) := :new.rowid;
end;
```

Trečias trigeris padaro kilpą kiekvienai įterptai ar pakeistai eilutei. Paimame rowed ir tikriname ar atvirkštinis variantas egzistuoja.

```
Create or replace trigger asymmetric
after insert or update on taskrelation
declare
  a_raw taskrelation%ROWTYPE;
  invalid_record exception;
begin
  for i in 1 .. state_pkg.newRows.count loop
    select t.* into a_raw
    from taskrelation t
    where t.rowid = state_pkg.newRows(i);

    select t.aftertaskid, t.beforetaskid into a_raw
    from taskrelation t
    where t.aftertaskid = a_raw.beforetaskID
    and t.beforetaskid = a_raw.aftertaskID;

    if a_raw.aftertaskid is not null then
      RAISE Invalid_record;
     end if;
  end loop;
   EXCEPTION
     WHEN Invalid_record THEN
```

```
      RAISE_APPLICATION_ERROR (
        num=> -20107,
        msg=> ,Asimetrinis apribojimas pazeistas!‘);
      WHEN no_data_found THEN null;
end;
```

“Antisymmetric” apribojimas neleidžia atvirkščio ryšio taip kaip “asymmetric”, bet skirtingai nuo jo leidžia tam pačiam atvejui dalyvauti abejose apribojimo rolėse. Taigi realizavimas yra panašus į “asymmetric” apribojimo. Skiriasi tik trečio trigerio sąlyga, išsaukianti konfliktą.

```
Create or replace trigger antisymmetric
after insert or update on taskrelation
declare
  a_raw taskrelation%ROWTYPE;
  invalid_record exception;
begin
  for i in 1 .. state_pkg.newRows.count loop
    select t.* into a_raw
    from taskrelation t
    where t.rowid = state_pkg.newRows(i);

    select t.aftertaskid, t.beforetaskid into a_raw
    from taskrelation t
    where t.aftertaskid = a_raw.beforetaskID
    and t.beforetaskid = a_raw.aftertaskID;

    if a_raw.aftertaskid is not null and (a_raw.aftertaskid <> b_raw.beforetaskid) then
      RAISE Invalid_record;
     end if;
  end loop;
   EXCEPTION
     WHEN Invalid_record THEN
       RAISE_APPLICATION_ERROR (
         num=> -20107,
         msg=> ,Antisymmetric apribojimas pazeistas!‘);
      WHEN no_data_found THEN null;
end;
```

“Acyclic” apribojimo trečiasis trigeris yra skirtingas. Jis turi išvesti klaidą jei įterptas įrašas formuoja ciklą.

```
Create or replace trigger acyclic
after insert or update on taskrelation
declare
  a_raw taskrelation%ROWTYPE;
  invalid_record exception;
begin
  for i in 1 .. state_pkg.newRows.count loop
    select t.* into a_raw
    from taskrelation t
    where t.rowid = state_pkg.newRows(i);

    select a_raw.beforetaskid into a_raw.beforetaskid
    from dual
```

```
   where a_raw.beforetaskid IN
   (select t.aftertaskid
    from (select tr.aftertaskid, tr.beforetaskid
          from taskrelation tr
          minus
          select a_raw.aftertaskid, a_raw.beforetaskid from dual) t
    start with t.beforetaskid = a_raw.aftertaskid
    connect by prior t.aftertaskid = t.beforetaskid);

   if a_raw.beforetaskid is not null then

     RAISE Invalid_record;

   end if;
  end loop;

  EXCEPTION
    WHEN Invalid_record THEN
      RAISE_APPLICATION_ERROR (
        num=> -20107,
        msg=> ,Acyclic apribojimas pazeistas!‘);

    WHEN no_data_found THEN null;
end;
```

“Intransitive” apribojimo trečiasis trigeris yra skirtingas. Jis turi išvesti klaidą jei įterptas įrašas pereinamas.

```
Create or replace trigger Intransitive
after insert or update on taskrelation
declare
  a_raw taskrelation%ROWTYPE;
  invalid_record exception;
begin
  for i in 1 .. state_pkg.newRows.count loop
    select t.* into a_raw
    from taskrelation t
    where t.rowid = state_pkg.newRows(i);

    select a_raw.beforetaskid into a_raw.beforetaskid
    from dual
    where a_raw.aftertaskid IN
    (select t.aftertaskid
     from (select tr.aftertaskid, tr.beforetaskid
           from taskrelation tr
           minus
           select a_raw.aftertaskid, a_raw.beforetaskid from dual) t
     start with t.beforetaskid = a_raw.beforetaskid
     connect by prior t.aftertaskid = t.beforetaskid);

    if a_raw.beforetaskid is not null then
       RAISE Invalid_record;
    end if;
  end loop;
```

```
  EXCEPTION
    WHEN Invalid_record THEN
     RAISE_APPLICATION_ERROR (
       num=> -20107,
       msg=> ‚Intransitive apribojimas pazeistas!‘);

    WHEN no_data_found THEN null;
end;
```

Vientisumo apribojimai objektų aibėms (Integrity constraints on sets of objects).

***Lygybės apribojimas nebūtiniems ryšiams ar atributams (*Equality constraint on groups of optional relationships or attributes)**

Vykdomas check apribojimų realizavimu lentelės stulpeliams.

```
ALTER TABLE <lenteles_pav>
ADD CONSTRAINT <apriboijimo_pav>
        CHECK (
        (<stulpelio_pav> IS NOT NULL AND < stulpelio_pav > IS NOT NULL ... < stulpelio_pav > IS NOT NULL)
        OR
        (<stulpelio_pav > IS NULL AND < stulpelio_pav > IS NULL ... < stulpelio_pav > IS NULL));
```

Lygybės apribojimo pavyzdys lentelės Person stulpeliams sex ir birthday:

```
ALTER TABLE Person
ADD CONSTRAINT equality
CHECK ((sex IS NOT NULL AND birthday IS NOT NULL) OR
                (sex IS NULL AND birthday IS NULL));
```

***Poaibio apribojimas nebūtiniems atributams (*Subset constraint on optional attributes)**

Naudojamas nebūtiniems esybių atributams ir ryšiams gali būti realizuojamas naudojant check apribojimus.

Tokiu atvejų mes turime įvardinti visus leidžiamus variantus firsName ir lastName:

Pavyzdys apribojimus realizuojant check apribojimais:

```
ALTER TABLE Person
ADD CONSTRAINT subset
CHECK (
        (firstName IS NOT NULL AND lastName IS NOT NULL)
        OR
        (firstName IS NULL AND lastName IS NOT NULL)
        OR
        (firstName IS NULL AND lastName IS NULL));
```

Lentelė	Atributai
Person	firstName, lastName

firstName	lastName	Gautas rezultatas	Laukiamas rezultatas	Atitinka
Pertas	Salvis	Sistema leido įvesti, redaguoti duomenis	Įterpimo, koregavimo įvykdymas	√
null	null	Sistema leido įvesti, redaguoti duomenis	Įterpimo, koregavimo įvykdymas	√
Pertas	null	ORA-02290: check constraint (LABAS.SUBSET) violated	Pranešimas Sisteminis, kad pažeistas apribojimas	√
null	Salvis	Sistema leido įvesti, redaguoti duomenis	Įterpimo, koregavimo įvykdymas	√

47 pav.

Kita galimybė poaibio apribojimo vykdymo, realizacija naudojant trigerį:

```
CREATE OR REPLACE TRIGGER poaibio
  BEFORE INSERT OR UPDATE OF firstName, lastName on Person
  ore ach row
DECLARE
  Invalid_record EXCEPTION;
BEGIN
  IF :new.firstName IS NOT NULL AND :new.lastName IS NULL THEN
    RAISE Invalid_record;
  END IF;

  EXCEPTION
     WHEN Invalid_record THEN
      RAISE_APPLICATION_ERROR (
        num=> -20107,
        msg=> ‘Poaibio aprbojimas pazeistas!‘);
END poaibio;
```

Lentelė	Atributai
Person	firstName, lastName

firstName	lastName	Gautas rezultatas	Laukiamas rezultatas	Atitinka
Pertas	Kleiza	Sistema leido įvesti, redaguoti duomenis	Įterpimo, koregavimo įvykdymas	√
null	null	Sistema leido įvesti, redaguoti duomenis	Įterpimo, koregavimo įvykdymas	√

Pertas	null	ORA-20107: Poaibio aprbojimas pazeistas!	Pranešimas kad pažeistas poaibio apribojimas	√
null	Salvis	Sistema leido įvesti, redaguoti duomenis	Įterpimo, koregavimo įvykdymas	√

48 pav.

Poaibio apribojimas nebūtiniems ryšiams* (Subset constraint on optional relationships)*:

```
CREATE OR REPLACE TRIGGER subset
  BEFORE UPDATE on B
  ore ach row
DECLARE
   b_raw A%ROWTYPE;
BEGIN
  select B.* into a_raw
  from B
  where B.ID = :new.id
  and   B.AID = :new.aid;
  EXCEPTION
     WHEN NO_DATA_FOUND THEN
      RAISE_APPLICATION_ERROR (
        num=> -20107,
        msg=> Poaibio apribojimas pazeistas!‘);
END subset;
```

Išskyrimo apribojimas *nebūtiniems atributams* (Exclusion constraint on optional attributes)

```
        ALTER TABLE <lenteles_pav>
ADD CONSTRAINT <apribojimo_pav>
        CHECK (
        (<stulpelio_pav> IS NULL AND < stulpelio_pav > IS NULL ... < stulpelio_pav > IS NULL)
        OR
        (<stulpelio_pav > IS NULL AND < stulpelio_pav > IS NOT NULL ... < stulpelio_pav > IS NOT NULL)
        OR
        (<stulpelio_pav > IS NOT NULL AND < stulpelio_pav > IS NULL ... < stulpelio_pav > IS NOT NULL)
        . . . OR
        (<stulpelio_pav > IS NOT NULL AND < stulpelio_pav > IS NOT NULL ... < stulpelio_pav > IS NULL)
        );
```

Panaudosime šį apribojimą lentelės Person laukams managerID and employeeID užtikrinti, kad vienas ir tas pats asmuo gali būti darbuotojas arba vadybininkas, bet ne abu kartu.

```
Alter table Person
  add constraint exclusion
  check ((managerID is not null and employeeID is null)
                OR managerID is null and employeeID is not null)
OR (managerID is null and employeeID is null));
```

Lentelė	Atributai
Person	managerID, employeeID

employeeID	managerID	Gautas rezultatas	Laukiamas rezultatas	Atitinka
2	8	ORA-02290: check constraint (LABAS.EXCLUSION) violated	Pranešimas Sisteminis, kad apribojimas	√
null	null	Sistema leido įvesti, redaguoti duomenis	Įterpimo, koregavimo įvykdymas	√
3	null	Sistema leido įvesti, redaguoti duomenis	Įterpimo, koregavimo įvykdymas	√
null	8	Sistema leido įvesti, redaguoti duomenis	Įterpimo, koregavimo įvykdymas	√

49 pav.

Išskyrimo apribojimas ryšiams **(Exclusion constraint on optional attribute)**

```
CREATE OR REPLACE TRIGGER exclusion
  BEFORE INSERT on knygos_recenzentai
  for each row
DECLARE
   a_exists number := 0;
   Invalid_reviewer EXCEPTION;
BEGIN
  select count(k.autorius) into a_exists
  from knygos_autoriai k
  where k.knyga = :new.knyga
  and   k.autorius = :new.recenzentas;

  IF a_exists > 0 THEN
     RAISE Invalid_reviewer;
  END IF;
  EXCEPTION
     WHEN Invalid_reviewer THEN
       RAISE_APPLICATION_ERROR (
         num=> -20107,
         msg=> 'Recenzentas yra knygos autorius !');
END poaibio;
```

6 Išvados

- Palyginti įvairiuose modeliavimo metuoduose (ER, EER, HERM, ORM, UML, xUML, RDBMS) naudojami apribojimai.
- Išanalizuotos apribojimų specifikacijos ir jų realizavimo galimybės.
- Pateikti konceptualūs apribojimų modeliai.
- Apribojimų realizavimui pasirinkta ORACLE 10g DBVS.
- Išnagrinėta ir aprašyta ORACLE trigerių sintaksė, bei veikimo principai.
- Aprašytis visi vientisumo apribojimai, pateiktas jų vaizdavimas UML, bei jų kodas ORACLE sistemoje.
- Realizuoti visų tipų apribojimai duomenų bazėje.
- Pateikta metodinė medžiaga padėsianti vartotojui įsisavinti apribojimų veikimo principus jų realizavimo galimybes.
- Ištestuoti visi realizuoti vientisumo apribojimai.
- Sukurta sistema pateikianti informaciją apie apribojimus, jų šablonus. Kai kuriems apribojimams sistema pati generuoja PL/SQL kodą vartotojui pasirinkis atitinkamus parametrus.

7 Literatūra:

[1] Demuth, B., Hussmann, H., 1999. Using UML/OCL Constraints for Relational Database Design. France, R., Rumpe, B

[2] YUAN SHI INTEGRATED VERIFICATION OF CONSTRAINTS AND EVENT-AND-ACTIONORIENTED BUSINESS RULES, UNIVERSITY OF FLORIDA 2001

[3] ISO/IEC 9075:1999. Information Technology – Database Languages – SQL – part 2: Fondation. 1999.

[4] S Ceri and J. Widom. Managing Semantic Heterogeneity With Production Rules And Persistent Queues. In *Proceedings of the Nineteenth International Conference on Very Large Data Bases*

[5] O. Diaz, M. Piattini, C. Calero, Measuring Triggering-Interaction Complexity on Active Databases, *Information systems,* Vol. 26, No. 1, p. 15-34, 2001.

[6] Paton NW (1999). Active Rules in Database Systems. Springer, New York.

[7] Kudakwashe Dube[1], Bing Wu[1] and Jane Grimson, Framework and Architecture for the Management of Event- Condition-Action (ECA) Rule-Based Clinical Protocols *School of Computing, Dublin Institute of Technology, IrelandDepartment of Computer Science, Trinity College Dublin, Ireland, 2002*

[8] Bing Wu, Kudakwashe Dube, PLAN: a Framework and Specification Language with an Event-Condition- Action (ECA) Mechanism for Clinical Test Request Protocols, 2001

[9] Dongwon Lee[1],Wenlei Mao[2], Henry Chiu[3], Wesley W. Chu[2] Designing Triggers with Trigger-By-Example, Springer-Verlag London Ltd. ∧ 2004
Knowledge and Information Systems (2004)

[10] Chays D, Deng Y, Frankl PG, Dan S, Vokolos FI, Weyuker EJ. AGENDA: A test generator for database applications. *Technical Report TR-CIS-2002-04*, Department of Computer Science, Polytechnic University, Brooklyn, New York, 2002.

[11] Elmasri R, Navathe SB. *Fundamentals of Database Systems* (3rd edn). Addison-Wesley: Boston, MA, 2000.

[12] Cochrane R, Pirahesh H, Mattos N (1996) Integrating Triggers and Declarative Constraints in SQL Database Systems. VLDB. Mumbai (Bombay), India

[13] Agrawal R, Gehani N (1989) Ode (Object Database and Environment): The Language and the Data Model. ACM SIGMOD. Portland, OR

[14] Rosenwald, G.W., and Liu, C.C., "Rule-based System Validation through Automatic Identification of Equivalence Classes," *IEEE Transactions on Knowledge and Data Engineering*, vol. 9, no. 1, Jan./Feb., 1997, pp. 24-31.

[15] Ross, R., "The Business Rule Book: Classifying, Defining and Modeling Rules," Second Edition, *Business Rule Solutions*, LLC, Houston, TX, 1997.

[16] Rousset, M.C., "On the consistency of knowledge bases: The COVADIS system," *Computant. Intelligent*, vol. 4, 1988, pp. 166-170.