



VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

FUNDAMENTINIŲ MOKSLŲ FAKULTETAS

INFORMACINIŲ SISTEMŲ KATEDRA

Asta Navalinskaitė

**VERSLO TAISYKLIŲ IŠGAVIMO IŠ DUOMENŲ BAZIŲ TRIGERIŲ
METODAS**

**A METHOD FOR EXTRACTION OF BUSINESS RULES FROM DATABASE
TRIGGERS**

Baigiamasis magistro darbas

Informacinių sistemų programų inžinerija, 621E15002

Informatikos inžinerija

Vilnius, 2013

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

FUNDAMENTINIŲ MOKSLŲ FAKULTETAS

INFORMACINIŲ SISTEMŲ KATEDRA

TVIRTINU
Katedros vedėjas

(Parašas)
_____Dalius Mažeika_____
(Vardas, pavardė)

(Data)

Asta Navalinskaitė

**VERSLO TAISYKLIŲ IŠGAVIMO IŠ DUOMENŲ BAZIŲ TRIGERIŲ
METODAS**

**A METHOD FOR EXTRACTION OF BUSINESS RULES FROM DATABASE
TRIGGERS**

Baigiamasis magistro darbas

Informacinių sistemų programų inžinerija, 621E15002

Informatikos inžinerija

Vadovas

____ prof. dr. Olegas Vasilecas ____

(Moksl. laipsnis/pedag. vardas, vardas, pavardė)

(Parašas)

(Data)

Konsultantas

____ doktorantas Kęstutis Normantas ____

(Moksl. laipsnis/pedag. vardas, vardas, pavardė)

(Parašas)

(Data)

Lietuvių kalbos konsultantas ____ dr. doc. Angelė Kaulakienė ____

(Moksl. laipsnis/pedag. vardas, vardas, pavardė)

(Parašas)

(Data)

Vilnius, 2013

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
FUNDAMENTINIŲ MOKSLŲ FAKULTETAS
INFORMACINIŲ SISTEMŲ KATEDRA

Informatikos inžinerija

Informacinių sistemų programų inžinerija, 621E15002

TVIRTINU
Katedros vedėjas

(Parašas)

Dalius Mažeika

(Vardas, pavardė)

(Data)

BAIGIAMOJO MAGISTRO DARBO

UŽDUOTIS

.....Nr.

Vilnius

Studentei Astai Navalinskaitei
(Vardas, pavardė)

Baigiamojo darbo tema: Verslo taisyklių išgavimo iš duomenų bazių trigerių metodas
.....

patvirtinta 201...m. d. dekanų potvarkiu Nr.

Baigiamojo darbo užbaigimo terminas 201...m. d.

BAIGIAMOJO DARBO UŽDUOTIS:

1. Išnagrinėti taisyklių įgyvendinimo duomenų bazių valdymo sistemose principus.
2. Išanalizuoti DBVS įgyvendintų taisyklių išgavimo metodus.
3. Įvertinti ir pasirinkti taisyklių išgavimo metodą.
4. Suformuluoti reikalavimus pasirinktam taisyklių išgavimui iš DBVS trigerių.
5. Remiantis suformuluotais reikalavimais, pasiūlyti taisyklių išgavimo metodo praplėtimą bei jį palaikančios priemonės prototipo architektūrą.
6. Atlikti pasiūlyto taisyklių išgavimo metodo tyrimus bei nustatyti jo efektyvumą.

Baigiamojo darbo rengimo konsultantai:

doktorantas Kęstutis Normantas
(Moksl. laipsnis/pedag. vardas, vardas, pavardė)

Vadovas
(Paraša)

.....prof. dr. Olegas Vasilecas.....
(Moksl. laipsnis/pedag. vardas, vardas, pavardė)

Užduotį gavau

.....
(Parašas)

...Asta Navalinskaitė.....
(Vardas, pavardė)

.....
(Data)

Vilniaus Gedimino technikos universiteto
egzaminų, sesijų ir baigiamųjų darbų
rengimo bei gynimo organizavimo tvarkos
aprašo 2011-2012 m. m.

1 priedas

(Baigiamojo darbo sąžiningumo deklaracijos forma)

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

Asta Navalinskaitė, 20076332

(Studento vardas ir pavardė, studento pažymėjimo Nr.)

Fundamentinių mokslų fakultetas

(Fakultetas)

Informacinių sistemų programų inžinerija, ISIfm-11

(Studijų programa, akademinė grupė)

**BAIGIAMOJO DARBO (PROJEKTO)
SĄŽININGUMO DEKLARACIJA**

2013 m. birželio 10 d.

Patvirtinu, kad mano baigiamasis darbas tema „Verslo taisyklių išgavimo iš duomenų bazių trigerių metodas“ patvirtintas 2012 m. spalio 24 d. dekanu potvarkiu Nr. 368fm, yra savarankiškai parašytas. Šiame darbe pateikta medžiaga nėra plagijuota. Tiesiogiai ar netiesiogiai panaudotos kitų šaltinių citatos pažymėtos literatūros nuorodose.

Prenkant ir įvertinant medžiagą bei rengiant baigiamąjį darbą, mane konsultavo mokslininkai ir specialistai: Kęstutis Normantas. Mano darbo vadovas prof. dr. Olegas Vasilecas.

Kitų asmenų indėlio į parengtą baigiamąjį darbą nėra. Jokių įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs (-usi).

(Parašas)

Asta Navalinskaitė

(Vardas ir pavardė)

Vilniaus Gedimino technikos universitetas Fundamentinių mokslų fakultetas Informacinių sistemų katedra	ISBN ISSN Egz. sk. Data-.....-.....								
<table style="width: 100%; border: none;"> <tr> <td style="width: 30%; padding-bottom: 5px;">Antrosios pakopos studijų</td> <td style="padding-bottom: 5px;">Informacinių sistemų programų inžinerijos programos baigiamasis darbas</td> </tr> <tr> <td style="padding-bottom: 5px;">Pavadinimas</td> <td style="padding-bottom: 5px;">Verslo taisyklių išgavimo iš duomenų bazių trigerių metodas</td> </tr> <tr> <td style="padding-bottom: 5px;">Autorius</td> <td style="padding-bottom: 5px;">Asta Navalinskaitė</td> </tr> <tr> <td style="padding-bottom: 5px;">Vadovas</td> <td style="padding-bottom: 5px;">prof. dr. Olegas Vasilecas</td> </tr> </table>		Antrosios pakopos studijų	Informacinių sistemų programų inžinerijos programos baigiamasis darbas	Pavadinimas	Verslo taisyklių išgavimo iš duomenų bazių trigerių metodas	Autorius	Asta Navalinskaitė	Vadovas	prof. dr. Olegas Vasilecas
Antrosios pakopos studijų	Informacinių sistemų programų inžinerijos programos baigiamasis darbas								
Pavadinimas	Verslo taisyklių išgavimo iš duomenų bazių trigerių metodas								
Autorius	Asta Navalinskaitė								
Vadovas	prof. dr. Olegas Vasilecas								
Kalba: lietuvių									
Anotacija Baigiamajame magistro darbe nagrinėjama pasenusių informacinių sistemų palaikymo ir plėtros problema. Darbu siekiama patobulinti verslo taisyklių išgavimo procesą iš egzistuojančių informacinių sistemų programų. Darbe atliekama verslo taisyklių išgavimo iš programų sistemų metodų analizė bei pasiūlomas verslo taisyklių išgavimo iš DB trigerių metodas, kuris remiasi verslo taisyklių šablonų paieška duomenų bazių trigerių kontrolės srauto grafuose. Darbe formaliai specifikuojamas šablonų paieškos algoritmas bei pasiūlomi 15 taisyklių šablonai, suprojektuojama metodą palaikančių priemonių architektūra bei specifikuojamas funkcionalumas. Atliekamas eksperimentas, kurio metu rankiniu būdu įvykdomi visi siūlomo metodo etapai. Eksperimento rezultatai parodė, jog, lyginant rezultatus su pasiūlyta šablonų aibe, gautas rezultatas yra teigiamas, todėl pasiūlytas metodas yra efektyvus bei gali būti įgyvendinamas ir vystomas toliau. Darbą sudaro 9 dalys: paveikslų sąrašas, lentelių sąrašas, santrumpos, įvadas, literatūros analizė, siūlomo metodo aprašymas, praktinė dalis, išvados ir rezultatai, literatūra. Darbo apimtis – 78 p. teksto be priedų, 14 iliustr., 34 lent., 29 bibliografiniai šaltiniai. Atskirai pridedami darbo priedai.									
Prasminiai žodžiai: duomenų bazių valdymo sistema, verslo taisyklė, verslo žodynas, trigeris, išgavimas, kontrolės srauto grafas, verslo taisyklės šablonas.									

Vilnius Gediminas Technical University Faculty of Fundamental Sciences Department of Information Systems	ISBN ISSN Copies No. Date*.....*.....												
<table style="width: 100%; border: none;"> <tr> <td style="width: 30%;">Master Degree Studies</td> <td>Information Systems Software Engineering study programme</td> <td>Final Thesis</td> </tr> <tr> <td>Title</td> <td colspan="2">A Method for Extraction of Business Rules from Database Triggers</td> </tr> <tr> <td>Author</td> <td colspan="2">Asta Navalinskaitė</td> </tr> <tr> <td>Academic supervisor</td> <td colspan="2">Prof Dr Olegas Vasilecas</td> </tr> </table>		Master Degree Studies	Information Systems Software Engineering study programme	Final Thesis	Title	A Method for Extraction of Business Rules from Database Triggers		Author	Asta Navalinskaitė		Academic supervisor	Prof Dr Olegas Vasilecas	
Master Degree Studies	Information Systems Software Engineering study programme	Final Thesis											
Title	A Method for Extraction of Business Rules from Database Triggers												
Author	Asta Navalinskaitė												
Academic supervisor	Prof Dr Olegas Vasilecas												
Thesis language: Lithuanian													
Annotation This thesis tackles the problem of support and development legacy information systems. The main goal of this work is to improve the process of business knowledge extraction from existing information systems. For this reason, the analysis of related methods is performed and a method for business rules extraction from database triggers is proposed. The method is based on rule patterns matching within control-flow graphs of DB triggers. The algorithm for searching rule patterns as well as 15 rule patterns are formally specified in the thesis. Moreover, the work propose an architecture of method supporting tool suite and provides specification of tools functionality. In order to evaluate the method, the experiment of manual application of the method for a set of database triggers is conducted. The results of experiment shows that the proposed method is efficient and therefore may be implemented and evolved in the future. Thesis consists of: 9 parts: list of figures, list of tables, abbreviation, introduction, analysis of the literature, description of the method proposed, practical part, conclusions nad results, references. Work size – 78 pg. text without appendixes, 14 illustrations, 34 table, 29 bibliographic sources. Appendixes are attached separately.													
Keywords: database management system, business rule, business vocabulary, trigger, extraction, control-flow graph, business rules pattern.													

TURINYS

PAVEIKSLŲ SĄRAŠAS	8
LENTELIŲ SĄRAŠAS	8
SANTRUMPOS.....	9
ĮVADAS	11
Tyrimo objektas	12
Darbo tikslas ir uždaviniai	12
Temos naujumas ir aktualumas.....	13
Tyrimo metodika.....	13
Mokslinė darbo vertė	14
Darbo rezultatai.....	14
Darbo struktūra	14
Darbo aprobacija	15
1. LITERATŪROS ANALIZĖ.....	16
1.1. Verslo taisyklių įgyvendinimo duomenų bazių valdymo sistemomis principai	16
1.1.1. Verslo taisyklių apibrėžimas.....	16
1.1.2. Verslo taisyklių realizavimas duomenų bazėse	17
1.2. Įgyvendinti žinių išgavimo būdai.....	25
1.2.1. Programos kodo pjaustymas	26
1.2.2. Duomenų atvirkštinė inžinerija.....	28
1.2.3. Verslo žinių išgavimas iš egzistuojančios duomenų bazės.....	31
1.2.4. Gramatikos modeliavimo kalba „Gra2Mol“	36
1.2.5. SQL kalbos apdorojimo įrankis „General SQL parser“	39
2. SIŪLOMO METODO APRAŠYMAS.....	41
2.1. Metodo reikalavimai	41
2.2. Verslo taisyklių išgavimo iš trigerių metodo aprašymas	42
2.3. Verslo žodyno sudarymas	44
2.4. Kontrolės srauto grafo sudarymas	50
2.5. Verslo taisyklių šablonų sudarymas.....	51
2.5.1. Verslo taisyklės šablonų tipai	51
2.5.1. Trigerio įvykio šablonų tipai.....	52
2.6. Verslo taisyklės šablono pritaikymo algoritmas	54
3. PRAKTINĖ DALIS	56
3.1. Metodą palaikančios priemonės prototipo funkcionalumo specifikacija.....	56
3.2. Prototipo architektūra.....	60
3.2.1. Modulių įgyvendinimas	62
3.3. Eksperimentinis metodo įvertinimas.....	69
REZULTATAI IR IŠVADOS	71
LITERATŪRA	73
PRIEDAI.....	76
A priedas. Verslo taisyklių šablonai	76

Šablonų aibė: (IF..) taisyklių tipas	76
Šablonų aibė: (WHILE..) taisyklių tipas	83
Šablonų aibė: (WHILE..IF..ELSE..) taisyklių tipas	86
B priedas. AST, CFG, VŽ ir VT šablonų tipų metamodelių susiejimas	90
C priedas. Verslo taisyklės šablono pritaikymo algoritmo pseudokodas	91

PAVEIKSLŲ SĄRAŠAS

1 pav. DML trigerio sintaksė	22
2 pav. Statinis programos kodo pjaustymas	27
3 pav. Duomenų atvirkštinės inžinerijos algoritmas	29
4 pav. Verslo žinių išgavimo procesas iš egzistuojančių programų sistemų	33
5 pav. Transformacijų karkasas	34
6 pav. „Gra2Mol“ vykdomas procesas	37
7 pav. „Gra2Mol“ modernizavimo įrankis	38
8 pav. Verslo taisyklių išgavimo iš trigerių veiklos diagrama	43
9 pav. Verslo taisyklių išgavimo iš pasenusios DBVS prototipo panaudos atvejų diagrama	57
10 pav. Verslo taisyklių išgavimo iš pasenusios DBVS prototipo architektūra	62
11 pav. Žodyno elementų išgavimas naudojant „General SQL Parser“ įrankį	63
12 pav. MS SQL Server 2008 trigerio kodo išskirstymas blokais	64
13 pav. Trigerio kodo atvaizdavimas kontrolės srauto grafu	65
14 pav. AST, CFG, VŽ ir VT šablonų tipų metamodelių susiejimas	90

LENTELIŲ SĄRAŠAS

1 lentelė. Trigerių sintaksės raktiniai terminai	22
2 lentelė. SQL sakinių tipų šablonai terminų ir faktų išgavimui	46
3 lentelė. Verslo taisyklių šablonų taisyklių tipų pavyzdžiai	52
4 lentelė. Trigerio įvykio raktinių žodžių aprašymai ir taikomi šablonai	53
5 lentelė. Trigerių įvykių jungtinės raktinių žodžių išraiškos ir šablonai	53
6 lentelė. Sudaryti AST užduoties detalizavimas	57
7 lentelė. Sudaryti CFG užduoties detalizavimas	58
8 lentelė. Apibrėžti verslo žodyno taisyklės užduoties detalizavimas	58
9 lentelė. Patvirtinti verslo žodyną užduoties detalizavimas	58
10 lentelė. Sudaryti verslo žodyno šablonus užduoties detalizavimas	59
11 lentelė. Sudaryti kandidatus į verslo taisykles užduoties detalizavimas	59
12 lentelė. Apdoroti kandidatus į verslo taisykles užduoties detalizavimas	59
13 lentelė. Dokumentuoti patvirtintas verslo taisykles	60
14 lentelė. Taisyklė, aprašanti verslo taisyklės šabloną	65
15 lentelė. Šablonų pritaikymas	66
16 lentelė. Šablonų pritaikymas atsižvelgiant į įvykį	67
17 lentelė. Verslo analitikui pateikiamas kandidatas į verslo taisyklę	67

18 lentelė. Eksperimentinio tyrimo rezultatai	69
19 lentelė. Kontrolės srauto grafo (IF..) verslo taisyklės šablonas T_1	76
20 lentelė. Kontrolės srauto grafo (IF..) verslo taisyklės šablonas T_2	77
21 lentelė. Kontrolės srauto grafo (IF..) verslo taisyklės šablonas T_3	78
22 lentelė. Kontrolės srauto grafo (IF..) verslo taisyklės šablonas T_4	78
23 lentelė. Kontrolės srauto grafo (IF..) verslo taisyklės šablonas T_5	79
24 lentelė. Kontrolės srauto grafo (IF..) verslo taisyklės šablonas T_6	80
25 lentelė. Kontrolės srauto grafo (IF..) verslo taisyklės šablonas T_7	81
26 lentelė. Kontrolės srauto grafo (IF..) verslo taisyklės šablonas T_8	81
27 lentelė. Kontrolės srauto grafo (IF..) verslo taisyklės šablonas T_9	82
28 lentelė. Kontrolės srauto grafo (WHILE..) verslo taisyklės šablonas T_10	83
29 lentelė. Kontrolės srauto grafo (WHILE..) verslo taisyklės šablonas T_11	84
30 lentelė. Kontrolės srauto grafo (WHILE..) verslo taisyklės šablonas T_12	85
31 lentelė. Kontrolės srauto grafo (WHILE..IF..ELSE..) verslo taisyklės šablonas T_13	86
32 lentelė. Kontrolės srauto grafo (WHILE..IF..ELSE..) verslo taisyklės šablonas T_14	87
33 lentelė. Kontrolės srauto grafo (WHILE..IF..ELSE..) verslo taisyklės šablonas T_15	89

SANTRUMPOS

AST – abstrakčios sintaksės medis (angl. *Abstract syntax tree*)

ASTM – abstrakčios sintaksės medžio metamodelis (angl. *Abstract syntax tree metamodel*)

CFG – kontrolės srauto grafas (angl. *Control-flow graph*)

DAI – duomenų atvirkštinė inžinerija (angl. *Data reverse engineering*)

DB – duomenų bazė (angl. *Database*)

DBVS – duomenų bazių valdymo sistema (angl. *Database management system*)

DCL – duomenų valdymo kalba (angl. *Data Control Language*)

DDL – duomenų apibrėžimo kalba (angl. *Data definition language*)

DML – duomenų apdorojimo kalba (angl. *Data manipulation language*)

ECA – įvykis, sąlyga, veiksmas (angl. *Event-Condition-Action*)

KDM – žinių atradimo metamodelis (angl. *Knowledge Discovery Metamodel*)

IS – informacinė sistema (angl. *Information system*)

SBVR – verslo žodyno ir taisyklių semantikos apibrėžimai (angl. *Semantics of Business Vocabulary and Rules*)

SQL – struktūrizuota užklausų kalba (angl. *Structured query language*)

Trigeris – vykdomoji procedūra, duomenų bazės lentelę ar konkretų elementą, susiejanti duomenų keitimo įvykį su konkrečiais veiksmais, kuriuos DBVS atlieka kaskart, kai tas įvykis atsitinka (angl. *Trigger*)

T-SQL – struktūrizuota užklausų kalba, kuri palaiko transakcijas (angl. *Transact-SQL*)

Pseudo – informatikoje naudojamas priešdėlis, reiškiantis įvardijamo dalyko netikrumą, tariamumą, melagingumą

VT – verslo taisyklė (angl. *Business rule*)

XML – bendros paskirties duomenų struktūrų bei jų turinio aprašomoji kalba, dažniausiai naudojama duomenų mainams tarp skirtingo tipo sistemų (angl. *Extensible Markup Language*)

IVADAS

Pasenusi informacinė sistema yra bet kuri informacinė sistema, kurią vis sunkiau modifikuoti ir pritaikyti nuolat besikeičiančioms verslo taisyklėms. Informacinėje sistemoje paprastai įgyvendintos visos verslo taisyklės reikalingos organizacinei veiklai.

Informacinių sistemų programų palaikymas ir plėtra yra sudėtingas ir daug resursų reikalaujantis procesas. Kuriant ir diegiant informacines sistemas (toliau – IS), palaikymo sąnaudos yra planuojamos visiškai mažos, nors, kaip teigiama (Laučius ir Vasilecas 2008), jos yra 2 - 4 kartus didesnės nei projektavimo (IS kūrimo) sąnaudos (kartais ir dar didesnės). Palaikymas sudaro nuo 50 % iki 80 % sistemos gyvavimo ciklo sąnaudų (Leintz ir Awanson 1980; Nelson 1996).

Visos didelės programos modifikuojamos, pakeitimai įkeliami į sistemą, todėl jų struktūra ima blogėti. Problemos yra sąlygojamos daugelio faktorių:

- Informacinių sistemų programos sukurtos sujungiant įvairias technologijas, vartojant skirtingas programavimo kalbas bei duomenų bazių valdymo sistemas (toliau – DBVS).
- Sistemos dažniausiai realizuotos su pasenusia technine įranga (pvz., standusis diskas), kuri yra lėta ir ją brangu išlaikyti.
- Naudotos technologijos reikalauja specifinių programavimo sprendimų, kurie yra dažniausiai suprantami juos priėmusiems programuotojams.
- Nepakankama ar nesama detali techninė sistemos specifikacija, be to, ir programiniame kode ne visuomet išlaikomas tvarkingas programavimo stilius, kuris galėtų palengvinti programų suvokimą bei suprasti jų palaikymo procesą.

Susidomėjimas verslo taisyklių išgavimu iš informacinių sistemų programų egzistuoja kelis dešimtmečius. Literatūroje pateikiama įvairių būdų ar pagrindinių principų taisyklėms išgauti (Wang *et al.* 2004; Vasilecas ir Normantas 2011) iš programinio kodo, tačiau tik nedaugelis autorių (Paradauskas ir Laurikaitis 2006; Izquierdo ir Molina 2010; Normantas ir Vasilecas 2013) pasiūlo metodus taisyklėms išgauti iš duomenų bazių

valdymo sistemų. Viena iš priežasčių yra duomenų bazių valdymo sistemų įvairovė bei specifika, kuri apsunkina universalaus būdo sukūrimą.

Darbu siekiama įvertinti sąnaudas, reikalingas DBVS įgyvendintai verslo logikai išgauti: kokie dalykai turėtų būti nagrinėjami analizuojant verslo taisyklių įgyvendinimą duomenų bazės trigeriuose, kaip tą analizę būtų galima automatizuoti ir realizuoti.

Tyrimo objektas

Darbe atliekamo tyrimo objektas yra verslo taisyklių išgavimas iš duomenų bazių valdymo sistemų: atpažinimas programos kode ir verslo taisyklių išgavimas iš DB scheme vykdomų trigerių. Magistro darbe DBVS įvairovė susiaurinama iki Microsoft SQL Server 2008 duomenų schemas ir trigeriuose realizuotų verslo taisyklių išgavimo.

Darbo tikslas ir uždaviniai

Darbo tikslas yra įvertinti esamas ir patobulinti pasirinktą verslo taisyklių išgavimo metodą iš duomenų bazių trigerių. Tikslui pasiekti keliama šie uždaviniai:

- Išnagrinėti taisyklių įgyvendinimo duomenų bazių valdymo sistemose principus.
- Išanalizuoti DBVS įgyvendintų taisyklių išgavimo metodus.
- Įvertinti ir pasirinkti taisyklių išgavimo metodą.
- Suformuluoti reikalavimus pasirinktam taisyklių išgavimui iš DBVS trigerių.
- Remiantis suformuluotais reikalavimais, pasiūlyti taisyklių išgavimo metodo praplėtimą bei jį palaikančios priemonės prototipo architektūrą.
- Atlikti pasiūlyto taisyklių išgavimo metodo tyrimus bei nustatyti jo efektyvumą.

Temos naujumas ir aktualumas

Šiais laikais, kai beveik visos veiklos yra kompiuterizuotos, nuolat atsiranda trikdžių, lemiančių įmonių biudžetą - nėra taip paprasta ir pigu informacinių sistemų programose įgyvendinti atsinaujinusias verslo taisykles. Tačiau tai neišvengiamas procesas ir kuo sistema senesnė ar didesnė, tuo daugiau laiko ir pinigų reikia skirti naujam funkcionalumui įgyvendinti. Visa tai dėl anksčiau minėtų priežasčių: įvairios technologijos bei skirtingos programavimo kalbos; specifiniai programuotojo sprendimai sunkiai suprantami kitam programuotojui dėl komentarų stokos ar kode neišlaikyto tvarkingo stiliaus; kartais nereikšmingai atrodantis bei praleistas dokumentacijos atnaujinimas neretai klaidina ir tikrai nepalengvina ieškant optimalaus sprendimo naujai verslo taisyklei įgyvendinti. Dėl šių faktorių kaskart vis sunkiau pigiai ir greitai atnaujinti programų sistemas. Todėl:

- Naujų taisyklių integracijai labai trukdo aiškių sąsajų nebuvimas – sistema per daug sudėtinga.
- Keičiantis verslo taisyklėms, egzistuojančios IS nebeatitinka naujų keliamų reikalavimų.
- Prarandamas vientisumas tarp reikalavimų specifikacijos ir jų įgyvendinimo informacinių sistemų programose.
- Pasenusias sistemas yra labai sunku praplėsti, arba kartais net neįmanoma.

Dėl to ženkliai išauga IS palaikymo ir plėtos sąnaudos. Vien skirtingų DB naudojimas sudaro nepalankias sąlygas sistemos atnaujinimui.

Tyrimo metodika

Magistro darbe siūloma verslo taisyklės išgauti iš DBVS trigerių atpažįstant kodo sintaksę, pritaikant kontrolės srauto grafus (angl. *Control-flow graph*) (toliau – CFG) ir suformuotus taisyklių tipų šablonus. Pritaikius šablonus, kandidatai į verslo taisyklės pateikiami verslo analitikui tolimesniam apdorojimui: taisyklės patvirtinimui arba atmetimui. Siūlomas kontrolės srautų grafų ir sukonstruotų šablonų pritaikymo metodas yra

eksperimentinis ir gali būti papildomas vadovaujantis įvairiais aspektais: keičiamas CFG grūdėtumas, praplečiama šablonų aibė ar kita.

Mokslinė darbo vertė

Pasiūlytas naujas metodas duomenų bazės trigeriuose realizuotų verslo taisyklių išgavimui panaudojant verslo taisyklių šablonų aibę. Metodo etapai dalinai automatizuoti, išskyrus taisyklių aprašymo bei gautų rezultatų vertinimo etapus.

Darbo rezultatai

- Darbo rezultatai teikia naudą tolimesniems duomenų bazės trigeriuose realizuotų verslo taisyklių išgavimo būdų tyrimams.
- Darbe įvertintos sąnaudos reikalingos DBVS įgyvendintai verslo logikai išgauti: kokie dalykai turėtų būti nagrinėjami analizuojant verslo taisyklių įgyvendinimą duomenų bazės trigeriuose, kaip tą analizę būtų galima automatizuoti ir realizuoti.
- Pateikta verslo taisyklių šablonų pavyzdžių aibė, kurių pagrindu galima ją plėsti ir pritaikyti įvairiems programuotojų praktikoje vartojamiems SQL sakinių atvejams.
- Prototipo įgyvendinimas teiktų finansinę naudą suinteresuotoms verslo šalims, kurių poreikis yra atnaujinti arba pakeisti pasenusią informacinių sistemų programą.

Darbo struktūra

Darbą sudaro įvadas, trys skyriai ir išvados.

1 skyriuje pateikti verslo taisyklių įgyvendinimo duomenų bazių valdymo sistemomis principai. Palyginti verslo taisyklių apibrėžimai. Trumpai supažindinama su verslo taisyklių realizavimu duomenų bazėse: SQL kalbos tipai, trigerių tipai. Išanalizuoti įgyvendinti žinių išgavimo būdai siekiant įvertinti ir pasirinkti tinkamą metodą.

2 skyriuje pateikti reikalavimai siūlomam metodui, verslo taisyklių išgavimo iš trigerių metodo aprašymas. Metodas susideda iš trijų pagrindinių dalių: verslo žodyno sudarymo, kontrolės srauto grafų sudarymo, remiantis duomenų srautų analizės apibrėžimais bei verslo taisyklių šablonų sudarymu įvairiems sakinių tipams. Taip pat pasiūlytas ir pseudokalba aprašytas algoritmas gebantis kontrolės srautų grafų taisyklių tipui pritaikyti verslo taisyklės šabloną.

3 skyriuje aprašytas realizuojamos priemonės prototipas. Pateikta prototipo funkcionalumo specifikacija, priemonės architektūra. Detalizuoti architektūroje išskirti, skirtingą funkcionalumą įgyvendinantys, moduliai. Pateiktas rankiniu būdu ištirtas pavyzdys bei eksperimentinis metodo įvertinimas lyginant rezultatus su pasiūlyta verslo taisyklių šablonų aibe.

Rezultatai ir išvados skyriuje detalizuotos visų skyrių bendrosios išvados.

Prieduose pateikti kontrolės srautų grafų pavyzdžiai bei VT šablonų taisyklių tipus atitinkantys CFG taisyklių tipai.

Darbo aprobacija

Magistro darbas buvo pristatytas Informacinių sistemų mokslo laboratorijos seminare.

1. LITERATŪROS ANALIZĖ

Verslo taisyklės (toliau – VT) sąvoka yra labai plačiai naudojama. Todėl šiame skyriuje pateiktas konkretus šios sąvokos paaiškinimas, keletas skirtingų jos apibrėžimų. Pateikiamas VT realizavimo trigeriais būdas duomenų bazių valdymo sistemose, SQL kalbos raktiniai žodžiai bei trigerių sintaksė. Išnagrinėtas moksliniame straipsnyje (Paradauskas ir Laurikaitis 2006) aprašytas metodas, aktualia verslo taisyklių išgavimo metodų tema. Detaliai aprašomi duomenų atvirkštinės inžinerijos (toliau – DAI) procesai bei jų rezultatai. Taip pat trumpai pristatomas įrankis (Izquierdo ir Molina 2010; Gra2Mol), gebantis išgauti verslo taisykles iš duomenų bazių schemose realizuotų trigerių.

1.1. Verslo taisyklių įgyvendinimo duomenų bazių valdymo sistemomis principai

1.1.1. Verslo taisyklių apibrėžimas

Informacinių sistemų modeliavimui ir apibrėžimui yra labai svarbi verslo taisyklės sąvoka. Verslo taisyklės yra naudojamos skirtingiems duomenų aspektams, apdorojimams ir duomenų procesams išreikšti (Paton 1999). Kaip teigia autorius straipsnyje (Badia 2006), turbūt dėl savo neformalių apibrėžimų verslo taisyklės yra menkai nagrinėjamos įvairiose studijų kryptyse. Yra tik keletas susitarimų, padedančių tiksliai apibrėžti VT, rašant reikalavimų specifikacijas: koncepciniai modeliai, esybių – ryšių modeliai ir UML notacija. Tačiau tikslų duomenų bazėse realizuojamų verslo taisyklių apibrėžtų susitarimų nėra. Dažniausiai reliacinėse duomenų bazėse verslo taisyklės yra įgyvendinamos trigeriais. Verslo taisyklės sąvoka yra labai universali ir gali būti įgyvendinta naudojant kitas priemones.

Remdamiesi skirtingais šaltiniais, autoriai VT apibrėžia įvairiai:

- Elgsenų arba sąlygų, kurios turi būti įvykdytos versle, paskelbimas (OMG 2002).

- Teiginys, kuris nusako arba apriboja kokį nors verslo aspektą, deklaruoja verslo struktūrą, kontroliuoja verslo procesus arba daro jiems kitokia įtaka (Kolber *et al.* 1997).
- Taisyklės sukurtos tiesiogiai vartojant terminus ir faktus (Ross 2003).
- Politinių reikalavimų paremti taisyklių rinkiniai, procedūros arba apibrėžimai, kuriais reglamentuotos organizacijos vykdo verslą sąveikaudamos su savo klientais ar partneriais (Adreescu ir Uta 2007).

1.1.2. Verslo taisyklių realizavimas duomenų bazėse

Akivaizdu, kad verslo taisyklių apibrėžimų yra įvairių ir jos atlieka daug funkcijų. Darbe nagrinėjamas vienas iš VT realizavimo būdų: duomenų bazių valdymo sistemoje realizuotais trigeriais. Autorius (Avdejenkov 2009) teigia, kad kalbant programuotojo kalba, trigeris – tai speciali procedūra, kuri susiejama su lentele ir iškviečiama įvykus konkrečiam įvykiui. Tokios taisyklės dar vadinamos ECA (angl. *Event-Condition-Action*) tipo taisyklėmis: įvykis, sąlyga ir veiksmas. Įvykis nusako taisyklės veikimo pradžią, sąlyga patikrinama taisyklės veikimo pradžioje, ir veiksmas yra vykdomas, kai įvykis ir sąlyga įvyksta teisingai.

1.1.2.1. SQL kalba

SQL kalba vartojama reliacinėms operacijoms aprašyti. Šioje kalboje išskiriamos trys sakinių grupės: duomenų apibrėžimo sakiniai, kitaip dar vadinami duomenų apibrėžimo kalba (angl. *Data Definition language (DDL)*); duomenų apdorojimo sakiniai (angl. *Data Manipulation Language (DML)*) ir duomenų valdymo sakiniai (angl. *Data Control Language (DCL)*), taigi $SQL = DDL + DML + DCL$.

DDL sakiniiais, be kitų veiksmų, kuriamos duomenų bazės ir lentelės. DDL sakiniuose vartojami šie raktiniai žodžiai:

- CREATE – sukurti įvairiems objektams, pavyzdžiui, lentelėms.
- DROP – sunaikinti nurodytus objektus.

- ALTER – pakeisti jau egzistuojančius objektus sistemos darbo metu.
- TRUNCATE – pašalinti visus objektus iš lentelės įskaitant tarpus.
- COMMENT – pridėti komentarams.
- RENAME – pakeisti objekto pavadinimui.

Kuriant (apibrėžiant) lentelę, būtina nurodyti lentelės vardą, jos stulpelių vardus ir stulpelių tipus. Stulpelio tipu nusakoma, kokios rūšies duomenys galės būti jame įrašomi. SQL priimtinos tokios duomenų rūšys: tekstiniai duomenys, skaičiai, dvejetainiai skaičiai, datos ir laikai. Kiekviena duomenų rūšis su tam tikrais fiziniais atributais (pvz., baitų skaičius, skiriamas reikšmei saugoti) nusako duomenų tipą. Duomenų tipo sąvoka yra reliacinės teorijos domeno sąvokos atitikmuo.

Dažnai realiai vartojamos informacijos duomenys yra nevisi arba jų visai nėra. Į tai atsižvelgiama įvedant informaciją į kompiuterį. Todėl kartais vartojama sąvoka „nulis“ (NULL). Tačiau tai ne visada padeda, nežinojimas ne visada reiškia nulį. Atsižvelgiant į reikšmingumo faktą, kad duomenų nėra, SQL kalboje kiekvienam duomenų tipui yra speciali reikšmė NULL. Ši reikšmė nurodo, kad duomenys yra nežinomi, šiuo metu dar neįvesti, jau pašalinti ar panašiai. Tai tarsi reikšmė, nurodanti reikšmės nebuvimą.

DML sakiniais formuojamos užklausos (angl. *query*) duomenims rasti, atnaujinti, šalinti arba įterpti. Vartojami šie raktiniai žodžiai:

- SELECT ... FROM ... WHERE ... – gauti duomenis iš duomenų bazės, nurodant papildomas sąlygas.
- INSERT INTO ... VALUES ... – įkelti duomenis į nurodytą lentelę.
- UPDATE ... SET ... WHERE ... – atnaujinti esamus duomenis lentelėje.
- DELETE FROM ... WHERE ... – ištrinti visiems įrašams iš lentelės nurodant papildomas sąlygas. Paliekamos tuščios ertmės (NULL).

DCL sakiniiais koordinuojamas visas vartotojų darbas su DB. Vartojami šie raktiniai žodžiai:

- GRANT – vartotojo prieigos teisės prie duomenų bazės.
- REVOKE – vartotojų teisių panaikinimas.

Transakcija – loginis duomenų bazių darbo vienetas, veiksmų duomenų bazėje seka. Transakcija vykdoma iki galo arba anuliuojama visiškai. Transakciją galima apibūdinti kaip duomenų bazės transformavimą iš vienos suderintos į kitą suderintą būseną, tačiau tarpiniuose taškuose duomenų bazė yra nesuderintos būsenos (Transakcija 2013). Duomenų bazių valdymo sistemose, kurios palaiko transakcijas (toliau – T-SQL) galima naudoti šias komandas:

- BEGIN – nurodoma pradėti atominę operaciją.
- COMMIT – patvirtinama sėkmingai baigiama transakcija.
- ROLLBACK – nurodoma, kad visa transakcija atšaukiama.

Nagrinėjamų SQL sakinių išsamią sintaksę sunku pateikti, nes neretai SQL sakinių sintaksė konkrečiuose DBVS skirtinga, taip pat ji skiriasi ir nuo standarto. Autorius straipsnyje (Dadashzadeh 2006) aptaria SQL standarto kilmę ir palygina kelias kalbas. Praktiškai SQL dialektų skaičius yra lygus platinamų DBVS skaičiui.

1.1.2.2.Trigeriai

Naudojant trigerius yra realizuojama dalis dalykinės srities verslo taisyklių. Apibrėždamas triggerį, vartotojas su konkrečiu lentelės duomenų keitimo įvykiu susieja konkrečius veiksmus, kuriuos duomenų bazės valdymo sistema atlieka kaskart, kai tas įvykis atsitinka. Yra trys duomenų keitimo įvykiai, ir juos atitinka trys duomenų apdorojimo sakiniai: INSERT, UPDATE ir DELETE (įrašyti, atnaujinti ir ištrinti). Veiksmai, kurie atliekami įvykus konkrečiam įvykiui, apibrėžti SQL sakiniiais.

Trigeriai kuriami sakiniu CREATE TRIGGER, kuriame nurodoma (Baronas 2002):

- Trigerio kvietimo momentas: prieš vykdant duomenų keitimą ar po jo.
- Duomenų keitimo įvykis (angl. *triggered action*) (INSERT, UPDATE ar DELETE), su kuriuo susiejamas trigeris.
- Trigerio pagrindas (angl. *trigger body*) – vienas ar keli SQL sakiniai, kuriuos reikia įvykdyti, kai trigeris išskviečiamas.
- Trigerio pagrindo vykdymo sąlyga: iškviesto pagrindo sakinius vykdyti besąlygiškai ar tik tuomet, kai patenkinta konkreti sąlyga.

MS SQL Server 2008 trigeriai

SQL Server 2008 kuriami DML (angl. *Data Manipulation Language*), DDL (angl. *Data Definition Language*) arba prisijungimo (angl. *Logon*) trigeriai. Trigeriai yra specialios rūšies saugomoji procedūra, kuri automatiškai paleidžiama, kai įvyksta įvykis su tam tikrais duomenimis, patalpintais duomenų bazėje.

- DML trigeriai vykdomi, kai vartotojas bando pakeisti duomenis per duomenų apdorojimo kalbos (DML) įvykį lentelėse arba rodiniuose. Šie trigeriai suveikia, kai bet koks galiojantis įvykis yra aktyvuojamas, nepriklausomai nuo to, ar visi lentelės įrašai paveikiami.
- DDL trigeriai vykdomi įvairių duomenų aprašymo kalbos (DDL) įvykių. Tokie įvykiai visų pirma atitinka T-SQL (angl. *Transact-SQL*) kalbos CREATE, ALTER, DROP sakinius, ir tam tikras sistemos saugojimo procedūras, vykdomas DDL operacijomis.
- Prisijungimo trigeriai suveikia prisijungimo (angl. *Login*) metu, kurie iškeliami pradedant vartotojo sesiją. Trigeriai gali būti sukurti tiesiogiai T-SQL kalbos sakiniiais arba panaudojant bibliotekos objektus (angl. *assembler*).

Trigerių paskirtys įvairios, tačiau dažniausiai naudojami atvejai:

- Audito trigeriai (duomenų keitimai fiksuojami audito lentelėse).

Audito informacija gali būti registruojama naudojant trigerius. Atnaujinant lentelės įrašą inicijuojamas audito trigerio veikimas, kai nurodoma, kokią informaciją įrašyti į auditui skirtą lentelę – dažniausiai tai būna prisijungusio naudotojo identifikuojantys duomenys, tiksli data bei atliktų pakeitimų duomenys. Tokius trigerius galima lengvai atpažinti: naudojami trigerio raktiniai žodžiai apibrėžti įvykiui ir trigerio kodo raktiniai žodžiai `INSERT INTO <audito lentelė> VALUES <visi stulpeliai ir sisteminė informacija – GETDATE(), USER_NAME()>`

- Verslo logiką įgyvendinantys trigeriai.

Tai trigeriai, kurie aktyvuojasi įvykus įvykiui ir atlieka verslo logiką atitinkančius veiksmus, aprašytus trigerio kode: įvairių sąlygų tikrinimai, naudų duomenų sukūrimas ar užpildymas, ribojimai, pranešimų rodmenys.

- Duomenų kopijavimo ar replikavimo trigeriai.
- Perteklinių duomenų spartai padidinti trigeriai.

DML trigerių sintaksė

Kadangi šio darbo nagrinėjamas objektas yra verslo taisyklės, tai toliau yra nagrinėjama DML trigerių sintaksė: raktiniai žodžiai, įvykiai, sąlygos ir veiksmai, kuriais manipuliuojant realizuojamos verslo taisyklės (Create trigger 2013). Taip pat DML trigeriai dažnai naudojami duomenų vientisumui užtikrinti. Lentelės arba rodinio (angl. view), įvykio INSERT, UPDATE arba DELETE, trigerių sintaksė pateikiame paveiksle 1.

```

CREATE TRIGGER [ schema_name . ]trigger_name
ON { table | view }
[ WITH <dml_trigger_option> [ ,...n ] ]
{ FOR | AFTER | INSTEAD OF }
{ [ INSERT ] [ , ] [ UPDATE ] [ , ] [ DELETE ] }
[ NOT FOR REPLICATION ]
AS { sql_statement [ ; ] [ ,...n ] | EXTERNAL NAME <method specifier [ ; ] > }

<dml_trigger_option> ::=
    [ ENCRYPTION ]
    [ EXECUTE AS Clause ]

<method_specifier> ::=
    assembly_name.class_name.method_name

```

1 pav. DML trigerio sintaksė
(Šaltinis: Create trigger 2013)

Jei egzistuoja apribojimai (SQL sąlyga) trigerio lentelėje, jie yra tikrinami po INSTEAD OF trigerio atlikimo arba prieš AFTER trigerio atlikimą:

- Jei apribojimai yra pažeidžiami, INSTEAD OF trigerio veiksmai grąžinami atgal (angl. *rolled back*) ir po AFTER trigeris neaktyvuojamas.
- AFTER trigeris yra įvykdomas tik po SQL sąlygos sėkmingo įvykdymo. Šis sėkmingas įvykdymas apima visus pakopinių (angl. *cascade*) nuorodų veiksmus ir apribojimų patikrinimus, susijusius su objektų atnaujinimu arba trynimu.

Lentelėje pateikiami trigerių sintaksėje vartojamų raktinių terminų aprašymai.

1 lentelė. Trigerių sintaksės raktiniai terminai

Terminas	Aprašymas
schema_name	Schemos pavadinimas, kuriai priklauso DML trigeris. DML trigeris apima pateiktą schemos lentelę arba rodinį. schema_name negali būti nurodyta DDL arba prisijungimo trigeriams.
trigger_name	Trigerio pavadinimas. trigger_name turi atitikti identifikavimo taisyklės, tokias kaip: trigger_name negali prasidėti simboliais # arba ##.

Terminas	Aprašymas
table view	Lentelė arba rodinys, kurioje DML trigeris yra vykdomas. Lentelės arba rodinio viso vardo nurodymas yra neprivalomas (angl. <i>optional</i>). Rodinys gali būti pateiktas tik INSTEAD OF trigeriui. DML trigeriai negali būti apibrėžti laikinose lentelėse.
ENCRYPTION	CREATE TRIGGER užšifravimo sakiny, kuris nurodo kopijavimą. WITH ENCRYPTION sakiniai negali būti naudojami CLR trigeriams.
EXECUTE AS	Nurodo saugumo kontekstą, pagal kurį vykdomas trigeris. Leidžia kontroliuoti vartotojo paskyrą, iš SQL Server esamų vartotojų, patvirtinant teises į bet kuriuos duomenų bazės objektus, kuriems yra nurodomi trigeriai.
FOR AFTER	AFTER nurodo, kad DML trigeris aktyvuojamas tik tada, kai visos operacijos, nurodytos triggerio SQL sakinyje, įvykdomos sėkmingai. Visi nuorodų pakopiškumo (angl. <i>cascade</i>) veiksmai arba apribojimai patikrinami. AFTER yra numatytasis; FOR yra nurodomas tik raktiniams žodžiams. AFTER trigeriai negali būti nustatomi rodiniais.
INSTEAD OF	Nurodo, kad DML trigeris vykdomas vietoj suveikusio (angl. <i>triggering</i>) SQL sakinio, todėl neatsižvelgiama į tame sakinyje apibrėžtus veiksmus. INSTEAD OF negali būti nurodomas DDL ar prisijungimo (angl. <i>Logon</i>) trigeriuose.
[DELETE] [,][INSERT] [,][UPDATE]	Duomenų pakeitimo raktiniai terminai, kurie aktyvuoja DML trigerius, kai yra bandoma atlikti veiksmus su nurodyta lentele arba rodiniu. Turi būti nurodomas bent vienas variantas. Triggerio apibrėžime leidžiamas bet kuris ar bet kokia tvarka derinys iš šių variantų. INSTEAD OF trigeriams, DELETE parinktis nėra leidžiama lentelėms, kurios turi nurodytų ryšių su pakopiniais (angl. <i>cascade</i>) veiksmiais ON DELETE . Panašiai UPDATE pasirinkimas neleidžiamas lentelėms, kurios turi nurodytų ryšių, su pakopiniais (angl. <i>cascade</i>) veiksmiais ON UPDATE.

Terminas	Aprašymas
NOT FOR REPLICATION	Nurodo, kad trigeris neturėtų būti vykdomas, kai kopijavimo agentas keičia lentelę, kuri nurodyta trigeryje (kai vykdomi replikavimo trigeriai).
sql_statement	Tai trigerio sąlygos ir veiksmai. Trigerio sąlygos apibrėžia papildomus kriterijus, kurie nustato, ar DML, DDL arba prisijungimo įvykiai gali sukelti trigerio veiksmų atlikimą. Trigerio veiksmai T-SQL sakiniuose įsigalioja, kai operacija patikrinta. Trigeriai gali turėti ne vieną ir bet kokios rūšies T-SQL sakinį su išimtimis. Trigeriu siekiama patikrinti ar pakeisti duomenis, pagrįstus duomenų pakeitimo ar apibrėžimo sakiniu; jis negrąžina duomenų vartotojui. T-SQL sakiniai trigeriuose dažnai priklauso nuo kalbos srauto kontrolės. DML trigeriai naudoja pašalintus duomenis ir įterpia į logines (konceptines) lenteles. Lentelės yra struktūriškai panašios. Pvz., imami visi duomenys iš trinamos lentelės ir talpinami į pildomąją lentelę: <pre>SELECT * FROM deleted</pre> Šios lentelės vadinamos laikinosiomis lentelėmis: <code>deleted</code> ir <code>inserted tables</code>. DBVS automatiškai sukuria ir tvarko šias lenteles. Galima naudoti šias laikinas lenteles patikrinti tam tikrų duomenų pakeitimų poveikį ir arba nustatyti sąlygas skirtas DML trigerio veiksams. Negalima tiesiogiai pakeisti lentelių duomenų, ar atlikti duomenų aprašymo kalbos (DDL) operacijų lentelėse, pvz., <code>CREATE INDEX</code>.
<method_specifier>	CLR trigeriui, nurodo, kurį bibliotekos objekto (angl. <i>essembler</i>) metodą susieti su trigeriu .

1.1.2.3. Trigeriais realizuotų verslo taisyklių privalumai ir trūkumai

Nors trigeriai duomenų bazių valdymo sistemose ir turi daug privalomų, tačiau jie nėra plačiai naudojami praktikoje. Autorius (Avdejenkov 2009) teigia, kad vieną ar kelias verslo taisykles galima lengvai sukurti, jos yra lengvai kontroliuojamos, derinamos ir testuojamos. Bet kai atsiranda daug taisyklių, jų palaikymas, derinimas ir testavimas tampa vis sudėtingesnis. Iškyla taisyklių galutinio užbaigimo, rekursijos fiksavimo, sudėtingumo nustatymo ir daugelis kitų problemų.

Pagrindinis trigerių privalumas – jais apibrėžtos dalykinės verslo taisyklės saugomos duomenų bazėje. Privalumai:

- Paspartina programavimą – jie įsimenami duomenų bazėje ir veiksmų nereikia kartoti kiekvienoje programoje.
- Palengvina dalykinių taisyklių laikymąsi – konkretus trigeris visuomet išskviečiamas reikiamu momentu.
- Yra globalūs – pasikeitus dalykinėms verslo taisyklėms, tereikia pakeisti triggerį viename egzemplioriuje, o ne visas programas, kuriose yra taisyklių užtikrinimo sakiniai.

Trigerių trūkumų:

- DB sudėtingumas – trigeriams tapus DB dalimi, ši tampa daug sudėtingesnė. Kai duomenų bazėje yra daug trigerių, tai netgi paprastos taikomosios programos sukūrimas gali būti keblus dėl sudėtingos trigerių veiksmų logikos.
- Paslėpta logika – kai dalykinės srities taisyklės yra slaptos duomenų bazėje, paprasti duomenų keitimo veiksmai gali sukelti sudėtingą duomenų analizę. Taikomosios programos kūrėjas neturi galimybės tvarkyti visus vykstančius duomenų bazės procesus, kadangi veiksmai gali sužadinti kitus, paslėptus veiksmus.
- Paslėpta įtaka našumui – kai bazėje yra trigerių, duomenų keitimas nėra skaidrus. Paprastas reikšmės keitimas gali sukelti daugelio duomenų apdorojimą, kuriam reikės daug laiko. Gali būti sunku suvokti, kodėl duomenų keitimas vyksta taip lėtai.

1.2. Įgyvendinti žinių išgavimo būdai

Žinių išgavimo (angl. *knowledge extraction*) tikslas yra pusiau automatinis žinių atradimas ir išgavimas iš įmonėje realizuotos pasenusios sistemos: sugeneruoti detalų sistemos kodo aprašymą įtraukiant subjektus ar esybes (angl. *entities*) ir ryšius tarp jų,

verslo taisyklės, duomenų formavimo bei apribojimo pranešimus ir kita. Visą šią informaciją vadiname verslo žiniomis (Paradauskas ir Laurikaitis 2006).

Yra pasiūlyta keletas metodų pasenusią sistemą modifikuoti arba pakeisti nauja. Šie metodai klasifikuojami į tris kategorijas – pertvarkymas (angl. *redevelopment*), apgaubimas (angl. *wrapping*), migravimas (angl. *migration*):

- Pertvarkymas apima tokius procesus kaip sistemos atnaujinimas nuo nulio – nauja techninės įrangos platforma (pvz., standusis diskas), architektūra, įrankiai ir duomenų bazės.
- Apgaubimui naudojamas papildomas programinės įrangos komponentas, vadinamas apgaubikliu (angl. *wrapper*), kuris papildo pasenusią sistemą – funkcionalumas nepakinta.
- Migravimas leidžia pasenusią sistemą perkelti į naują aplinką, kur ji tampa lengviau palaikoma ir pritaikoma naujiems verslo reikalavimams.

Šie metodai išlaiko originalų pasenusios informacinės sistemos funkcionalumą ir sukauptus duomenis, išvengiama visiško sistemos pakeitimo į naują.

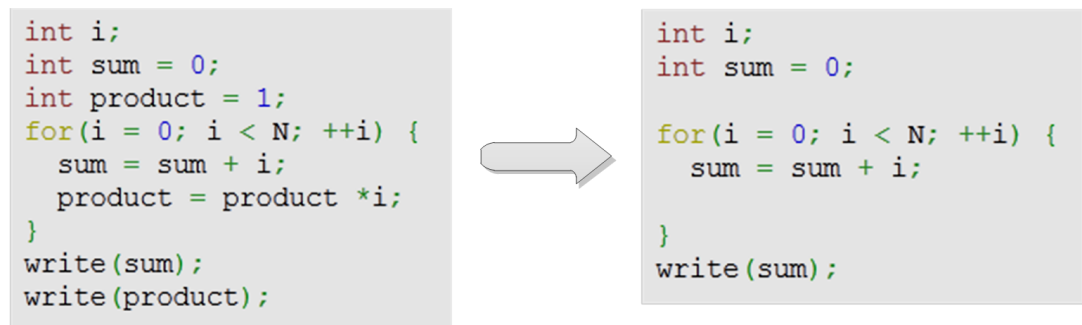
Pasenusios sistemos pakeitimo metodas dalinai sujungia visas tris ankščiau išvardytas kategorijas. Pirmasis etapas sprendžiant pasenusių sistemų problemas yra tikslus, detalus informacinės sistemos vaizdas. Dėl to reikia atlikti atvirkštinę inžineriją (angl. *reverse engineering*) ir atrasti bei išgauti visas, kiek įmanoma, verslo žinias (angl. *business knowledge*) iš pasenusios sistemos.

1.2.1. Programos kodo pjaustymas

Programos kodo pjaustymas (angl. *Program slicing*) yra aibė galimų technikų išgauti specifines ir tikslias žinias iš pasenusio kodo. Išgautos žinios atitinkamai vizualizuojamos ir pristatomos suteikiant pagrindą tolimesnei jų analizei, klasifikavimui ar pertvarkymui (Rodrigues ir Barbosa 2010). Kodo pjaustymas buvo pristatytas autoriaus Weiser'io straipsnyje (Weiser 1981) kaip technika, galinti suprasti ir suderinti programas. Nuo tada programos pjaustymas stipriai pasikeitė ir yra siūloma įvairių skirtingų pjaustymo

algoritmų. Yra išskiriami du tipai – statinis (angl. *static slicing*) ir dinaminis (angl. *dynamic slicing*) programos pjaustymas:

- Statinis pjaustymas remiasi Weiser'io pasiūlytu apibrėžimu. Statiniu pjaustymu yra apskaičiuojami ir surandami iš eilės einantys netiesiogiai susijusių sakinių rinkiniai, atsižvelgiant į duomenų ir kontrolės priklausomybes. Paveiksle 2 pavaizduotas statinis programos kodo pjaustymas atsižvelgiant į kriterijus (`write(sum)`, {`sum`}).



2 pav. Statinis programos kodo pjaustymas

- Dinaminis pjaustymas panaudoja vykdomosios programos informaciją: sugeba suvaldyti visus sakinius, kurie turi įtakos kintamojo reikšmei, vykdomus programos kode.

Statinio ir dinaminio pjaustymo skirtumui suvokti pateikiamas pavyzdys. Tarkim, yra programos kodas, kuriame yra iteracinis blokas, susidedantis iš IF – ELSE sakinių blokų. Yra keletas sakinių, lemiančių apibrėžtą kintamąjį ir IF, ir ELSE blokuose. Statinio pjaustymo metu visa programa yra peržvelgiama nepaisant tam tikrų vykdymo sąlygų – abiejuose IF ir ELSE blokuose yra išrenkami su kintamuoju susiję sakiniai. Dinaminio pjaustymo metu atsižvelgiama į vykdomųjų sakinių blokus – jei IF sakinyje yra tenkinama kintamojo sąlyga, tai ELSE sakinio blokas yra neličiamas.

1.2.1.1. Programos kodo pjaustymo metodo aprašymas

Straipsnyje (Wang *et al.* 2004) taikant programos pjaustymo metodą yra identifikuojami kintamieji ir suklasifikuojami pagal apibrėžtus būdus. Metodas taip pat pabrėžia kintamųjų sinonimų, pasitaikančių skirtinguose programos moduluose,

identifikavimo svarbą. Išgavus kintamuosius ir jų priklausomybes, kitas etapas yra duomenų analizė, kuri identifikuoja verslo elementus įgyvendintus „išpjautuose“ programos gabaluose (angl. *slice*). Verslo taisyklių rinkinys yra išgaunamas ir pateikiamas panaudojant skirtingus gautos informacijos peržiūros rodinius, kad būtų patikrintos ir patvirtintos verslo suinteresuotų šalių (pvz., pateikiami kodu arba grafu, kurie pristato priklausomybes tarp kodo blokų).

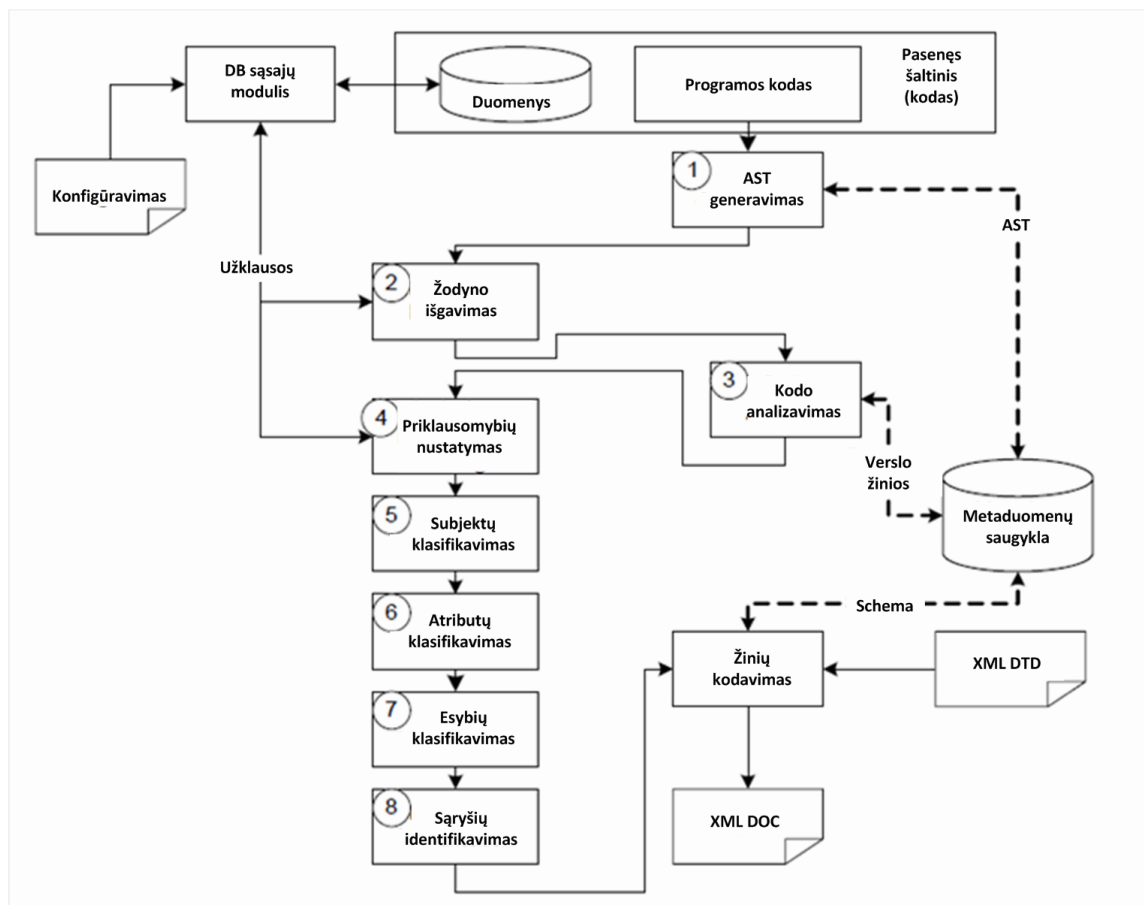
1.2.1.2.Privalumai ir trūkumai

Tai pusiau automatinis metodas realizuojamas panaudojant įrankius. Metodas nagrinėja visą programinį kodą išgaudamas kintamuosius, jų sąryšius, sinonimus bei verslo taisykles.

Šio metodo pagrindinis trūkumas, kad galutinio rezultato analizavimas reikalauja gero programinės įrangos supratimo technologiniu aspektu, todėl peržiūros rodiniai yra sunkiai suprantami verslo analitikams ar suinteresuotoms šalims. Taip pat metodas visiškai nenagrinėja trigeriuose realizuotų verslo taisyklių.

1.2.2. Duomenų atvirkštinė inžinerija

Duomenų atvirkštinė inžinerija apibrėžiama kaip analitinių technologijų taikomoji programinė įranga, skirta iš vienos ar daugiau pasenusių duomenų šaltinių sudaryti struktūrinę informaciją (tokią kaip terminų apibrėžimas, schemas apibrėžimas) iš pasenusio kodo, kad būtų pagerintas DB dizainas ar sudaryta trūkstama schemas dokumentacija. Šiame straipsnyje (Paradauskas ir Laurikaitis 2006) duomenų atvirkštinė inžinerija taikoma tik reliacinėms duomenų bazėms. Tačiau kadangi reliacinės DB turi tik ribotą semantinę išraišką, be schemas, duomenų atvirkštinės inžinerijos algoritmas generuoja ir pateikia esybių – ryšių diagramą, kuri nėra aiškiai apibrėžta pasenusioje schemoje (tačiau netiesiogiai egzistuoja).



3 pav. Duomenų atvirkštinės inžinerijos algoritmas
(Šaltinis: Paradauskas ir Laurikaitis 2006)

3 pav. pavaizduota žinių išgavimo schema iš pasenusių duomenų bazių valdymo sistemų. Žinių išgavimas iš DBVS yra neatsiejamas nuo visa realizuojančio programos kodo: įtakojantys raktai gali būti paslėpti verslo pranešimuose ar kokiuose kituose elektroninės laikmenos pranešimuose. Priminimai ar elektroniniai sistemos laiškai gali būti užkoduoti verslo duomenys.

1.2.2.1. Duomenų atvirkštinės inžinerijos algoritmo aprašymas

DAI algoritmas dalijamas į dvi dalis: schemas išgavimas (angl. *schema extraction*) ir semantinė analizė (angl. *semantic analysis*), kurios veikia sluoksniavimo būdu. Šios dvi algoritmo dalys sudarytos iš aštuonių etapų, atvaizduotų 3 pav.. Be šių aštuonių etapų, atvaizduotoje algoritmo architektūroje taip pat yra trys palaikantys (atraminiai) komponentai: konfigūruojamas duomenų bazių sąsajų modulis (angl. *Database Interface*

Module), žinių kodavimas (angl. *Knowledge Encoding*) ir metaduomenų saugykla (angl. *Metadata Repository*). Duomenų bazių sąsajų modulis pateikia pagrindinius pasenusius šaltinius. Pažymėtina, kad šis komponentas vienintelis konkrečių šaltinių komponentas architektūroje, galintis atlikti žinių išgavimą iš skirtingų šaltinių – tik šiame sąsajos modulyje atpažįstami ir gali būti pakeičiami skirtingi šaltiniai. Žinių kodavime pateikiamos išgautos žinios atvaizduojamos XML tipo dokumente. Metaduomenų saugykloje saugoma tarpinė informacija reikalinga algoritmams, įskaitant naudotojo įvestus parametrus ir kodo abstrakčios sintaksės medį (pavyzdžiui išgautą iš ankstesnių kreipimosi į kodą).

DAI proceso algoritme vykdomi aštuoni etapai:

- Abstrakčios sintaksės medžio generavimas (1) (toliau – AST) (angl. *Abstract syntax tree generation*) iš pasenusio kodo. Šis medis naudojamas semantinei kodo analizei (tai trečias etapas iš atvaizduotos DAI architektūros paveiksle 3). Šio generavimo objektas yra susietos medžio reikšmės su programos kintamaisiais. Pavyzdžiui, „string“ formatas įvesties ar išvesties sakiniuose turi semantinę informaciją, kuri gali būti susieta su kintamaisiais, naudojamais įvesties ar išvesties sakiniuose.
- Žodyno išgavimas (2) (angl. *Dictionary Extraction*). Išgaunami lentelių ir atributų pavadinimai, esantys pasenusiame kode.
- Kodo analizavimas (3) (angl. *Code Analysis*) – yra dvejopas: papildomas išgauto žodyno reikšmių vientisumas srities semantika ir identifikuojamos verslo taisyklės bei ribojimai, kurie nėra aiškiai saugomi duomenų bazėje, bet yra svarbūs atvirkštinės inžinerijos procese. Šis kodo analizavimo etapas paremtas programos supratimu iš kodo pjaustymo ir modelio mašinos konstatavimo.
- Priklausomybių nustatymas (4) (angl. *Inclusion Dependency Mining*). Po reliacinės schemos išgavimo kitas etapas yra identifikuoti ribojimus, tai padeda klasifikuoti sąryšinius subjektus, kurie naudojami apibrėžiant realaus pasaulio esybes ir sąryšius tarp jų.

- Subjektų klasifikavimo (5) (angl. *Relation Classification*) etapo tikslas identifikuoti skirtingus subjektų tipus. Kai kurie iš jų atitinka realaus pasaulio objektus, o kiti subjektai atsiranda, kai yra sąveikaujami. Visi duomenų bazėje esantys subjektai klasifikuojami į keturis tipus: stiprūs, reguliarūs, silpni ir specifiniai.
- Atributų klasifikavimas (6) (angl. *Attribute Classification*) pagal: pirminį raktą, išorinį raktą, atitikties (angl. *dangling*) raktą, bendrinį (angl. *general*) raktą ir rakto nebuvimą.
- Esybių klasifikavimo (7) (angl. *Entity identification*) etape stiprių (silpnų) esybių subjektai, apibrėžti nuo penkto proceso etapo, yra tiesiogiai konvertuojami į stiprias (silpnas) esybes.
- Sąryšių identifikavimas (8) (angl. *Relation Identification*) pagal (4) etape priklausomybių ir ribojimų nuostačius yra identifikuojami sąryšiai tarp esybių.

Duomenų atvirkštinės inžinerijos paskutinis etapas yra žinių pristatymas (angl. *Knowledge Representation*). Po aštuonių proceso etapų iš pasenusios duomenų bazės yra išgaunamos: subjektų klasifikacija ir pavadinimai, atributų pavadinimai, pirminiai ir išoriniai raktai, duomenų tipai, paprasti ribojimai ir aiškūs teiginiai, sąryšiai ir jų pagrindai.

1.2.2.2. Privalumai ir trūkumai

Tai pusiau automatinis metodas, realizuojamas panaudojant įrankius. Metodas nagrinėja duomenų bazės valdymo sistemos kodą išgaudamas verslo žinias: duomenų schemas elementus ir sąryšius tarp jų.

Šio metodo pagrindinis trūkumas, kad nagrinėdamas duomenų bazių valdymo sistemas visiškai nenagrinėja trigerių ir juose realizuotų verslo taisyklių.

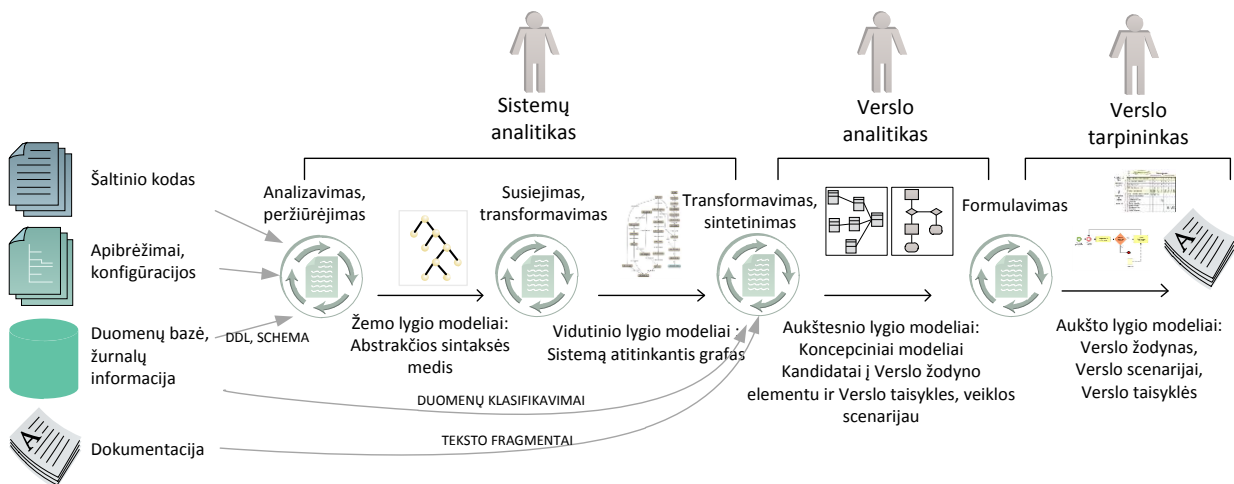
1.2.3. Verslo žinių išgavimas iš egzistuojančios duomenų bazės

Šiame skyriuje aptariamas verslo žinių išgavimo iš egzistuojančios DB straipsnis (Normantas ir Vasilecas 2013). Autorių teigimu vienas pagrindinių etapų išgaunant verslo

žinias įgyvendintas programų sistemose yra verslo žodyno išgavimas. Aprašomas procesas, išgaunantis žinių atradimo metamodelio (toliau – KDM) (angl. *Knowledge Discovery Metamodel*) standartus atitinkančią duomenų struktūrą, ir algoritmas, gebantis identifikuoti ir pateikti verslo terminų ir faktų kandidatus.

1.2.3.1. Verslo žinių išgavimo aprašymas

Verslo žinių išgavimas iš egzistuojančios programų sistemos remiasi atvirkštinės inžinerijos procesu. Procese yra įtraukiamos keletas pasikartojančių veiklų bei tas veiklas atliekančių vaidmenų: sistemų analitikas (angl. *System Analyst*), verslo analitikas (angl. *Business Analyst*), verslo tarpininkas (suinteresuotoji šalis) (angl. *Business Stakeholder*). Sistemų analitikas yra atsakingas už programinės įrangos parengimą ir pateikimą įvairiais skirtingais architektūriniais programinės įrangos aspektais. Programinis kodas, išteklių konfigūracijos, duomenų bazės schemas apibrėžimai yra analizuojami (angl. *parsed*) ir pateikiami abstrakčios sintaksės medžio pavidalu ir po to transformuojami į aibę galimų KDM modelių (OMG 2011). Šie modeliai tvarkomi taikant duomenų srautų analizės technikas (angl. *data flow analysis techniques*) (Khedker *et al.* 2009). Taip nustatomos priklausomybės tarp skirtingų sistemos komponentų. Nustačius priklausomybę, šie modeliai yra toliau tobulinami siekiant identifikuoti kandidatus į verslo žodyno įrašus (pvz., terminai, faktai), verslo veiklos scenarijus (pvz., panaudos atvejus (angl. *use-cases*)) ir verslo taisykles. Skirtingos faktų rūšys identifikuojamos atsižvelgiant į faktų tipų kategorijų nustatymo sistemą, apibrėžtą verslo žodyno ir taisyklių semantikos apibrėžimuose (angl. *Semantics of Business Vocabulary and Rules*) (OMG 2008). Verslo analitikas peržiūri išgautas žinias ir naudodamas automatinį transformavimą suformuluoja žinias, kurias pateikia verslu suinteresuotoms šalims. Pateikiamos žinios yra: žodyno įrašai, taisyklių sakiniai, sprendinių lentelės (angl. *decision tables*), procesų modeliai. Verslo žinių išgavimo procesas pavaizduotas 4 pav..

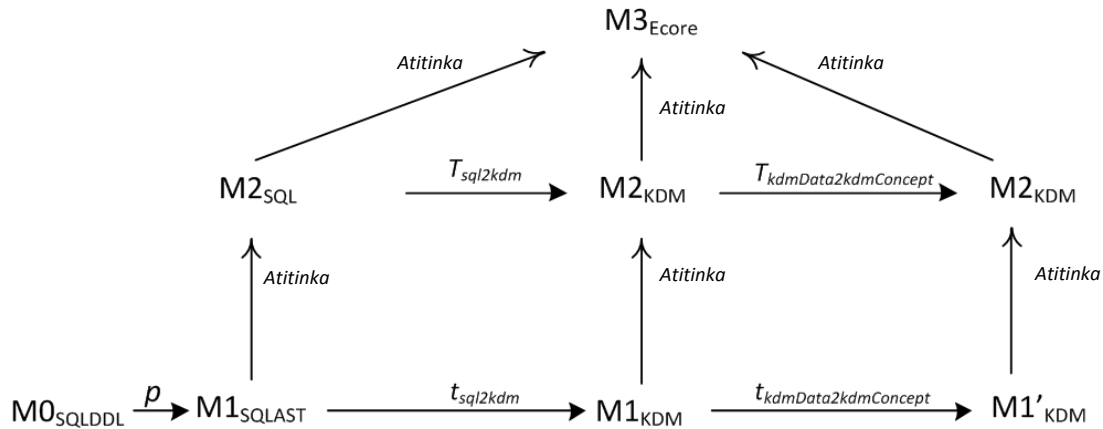


4 pav. Verslo žinių išgavimo procesas iš egzistuojančių programų sistemų
(Šaltinis: Normantas ir Vasilecas 2013)

Pagrindinių principų, verslo žodyno išgavimui iš egzistuojančios duomenų bazės schemas, etapai:

- Apibrėžiamas transformacijų karkasas (angl. *framework*), kuris naudojamas iš duomenų bazės struktūros apibrėžimų gauti KDM koncepcinį modelį (angl. *KDM Conceptual model*).
- Nustatomi SQL ir KDM duomenų modelio (angl. *KDM Data model*) elementų atitikmenys.
- Pateikiami KDM koncepcinio modelio pagrindiniai elementai, kurie gali reprezentuoti išgautą žodyną.

Transformacijų karkase, pavaizduotame 5 pav., pateikiamas verslo žinių, įgyvendintų programų sistemose, išgavimo modelis.



5 pav. Transformacijų karkasas
(Šaltinis: Normantas ir Vasilecas 2013)

Transformacijų karkasas susideda iš komponentų (nuo viršaus į apačią):

- $M3_{Ecore}$ – apibrėžimų ir modelių apdorojimų struktūra, dar kitaip vadinama metametamodeliu, gaunama naudojant „Eclipse“ modeliavimo įrankį (angl. *Eclipse modeling framework*).
- $M2_{SQL}$ – SQL abstrakčios sintaksės metamodelis naudojamas apibrėžti SQL DDL scenarijų gramatikos taisykles transformacijai į abstrakčios sintaksės medį.
- $M2_{KDM}$ – žinių atradimo (angl. *Knowledge Discovery*) metamodelis naudojamas egzistuojančių programų sistemų žinioms pateikti ir apdoroti.
- $M0_{SQLDDL}$ – SQL duomenų apibrėžimo kalbos (DDL) scenarijus. Tai yra pateikiamas šaltinis.
- $M1_{SQLAST}$ – pateikto duomenų šaltinio ($M0_{SQLDDL}$) abstrakčios sintaksės medžio modelis.
- $M1_{KDM}$ – KDM duomenų modelis, pristatantis $M1_{SQLAST}$.
- $M1'_{KDM}$ – KDM koncepcinis modelis, pristatantis verslo žodyną, išgautą iš $M1_{KDM}$.

- Tsql2kdm – aibė taisyklių atitikmenų (angl. *of mapping*) naudojamų SQL DDL scenarijų transformacijai į abstrakčią sintaksę.
- tsql2kdm – Tsql2kdm vykdymas.
- TkdmData2kdmConcept – aibė taisyklių atitikmenų (angl. *of mapping*) naudojamų išgauti terminų ir faktų vienetų iš KDM duomenų modelio (duomenų bazės duomenų struktūros).
- tkdmData2kdmConcept – TkdmData2kdmConcept vykdymas.
- p – apdorojant DDL scenarijus gaunamas M1SQLAST modelis; analizatorius yra sukurtas naudojant „XText“ analizatorių generatorius, kurie leidžia gramatikos „Ecore“ metamodelio taisyklės aprašyti norimu poskyriu (angl. *subsection*).

Verslo terminų ir faktų išgavimo algoritmas, skirtas verslo žodyno išgavimui iš KDM duomenų modelio išsaugant juos KDM koncepciniame modelyje. Naudojamas pseudokodo algoritmas užklausoms ir vertinimo modeliams apibrėžti. Yra vykdomi du pagrindiniai etapai:

- Pirmame etape peržiūrimi duomenų modelio elementai, identifikuojami terminai, priklausomybės (angl. *IsPropertyOf*) ir faktai (angl. *Assortment* tipo).
- Antrame etape analizuojami nuorodų ribojimai (angl. *referencing constrains*) ir identifikuojamos specializacijos (angl. *specialization*), ryšiai (angl. *Relationship*) ir faktai (angl. *Partitive* tipo).

Po atlikto eksperimentinio tyrimo straipsnyje (Normantas ir Vasilecas 2013) pateikti gauti rezultatai – teigiama, kad svarbą galima skirstyti dvejopai: išgautas žodynas leidžia analizuoti duomenų sąryšius; ir žodyno įrašai (faktai su susijusiais terminais) yra naudojami verslo scenarijų išgavimo kriterijoms konstruoti (pvz., kriterijai skirti kodo pjaustymo modeliui) bei verslo taisyklių išgavimui (pvz., kontroliniai sakiniai, kurie formuojami įvertinus ir pritaikius išgautus žodyno įrašus).

1.2.3.2.Privalumai ir trūkumai

Tai pusiau automatinis metodas, nagrinėjantis duomenų bazės šaltinio kodą ir išgaunantis verslo žinias. Procese aktyviai dalyvauja sistemos analitikas paruošdamas metamodelius reikalingus kodo apdorojimui. Verslo žodyno išgavimui naudojami įrankiai reikalauja gerų programavimo kalbos žinių. Atlikto eksperimento rezultatas yra išgautas verslo žodynas, kuris gali būti panaudojamas tolimesniam verslo taisyklių išgavimui.

Pagrindinis trūkumas – visiškai nenagrinėja trigeriuose realizuotų verslo taisyklių.

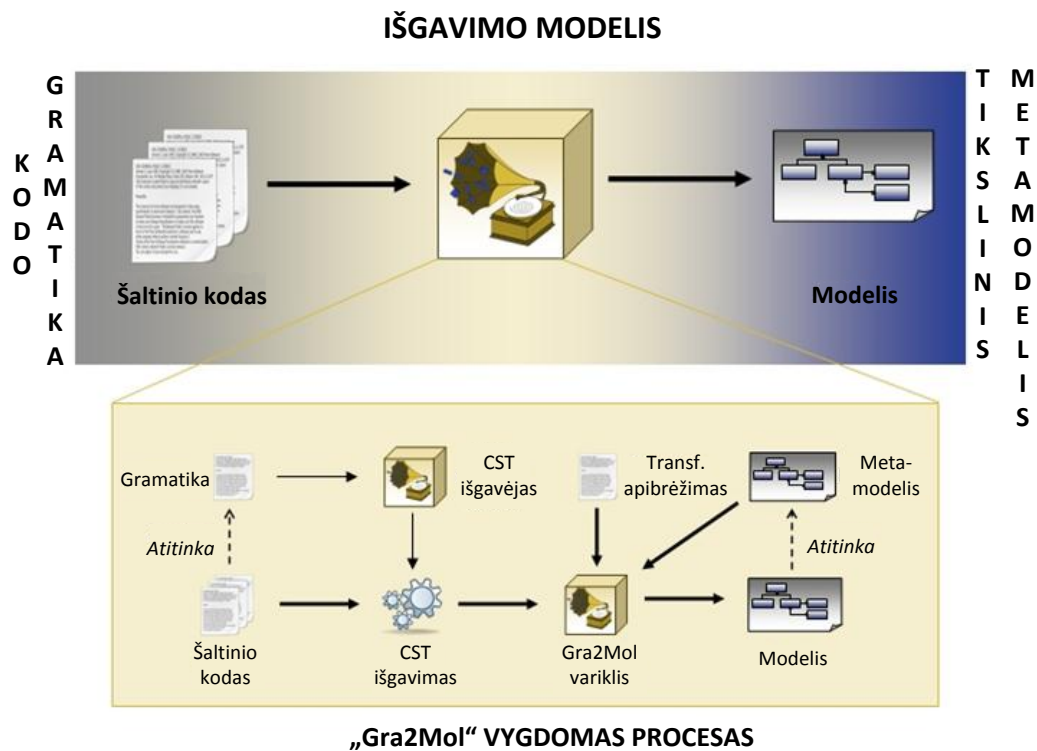
1.2.4. Gramatikos modeliavimo kalba „Gra2Mol“

Autoriai straipsnyje (Izquierdo ir Molina 2010) detaliai aprašo gramatikos modeliavimo kalbą „Gra2Mol“, apibrėžiančią ryšius tarp gramatikos elementų (šaltinio kodo) bei metamodelio elementų. Tai taisyklėmis grindžiama kalba, panaši į modelio transformacijos kalbą „RubyTL“ ar „ATL“. „Gra2MoL“, apibrėžia šaltinio elementą (gramatiką), tikslinį elementą (metamodelį) ir kai kuriuos metamodelio raktinius sakinius. Pagrindinis skirtumas nuo modelio transformacijos kalbos yra, kad „Gra2MoL“ yra orientuota į modelio išgavimą iš programos kodo. Be to, „Gra2MoL“ apibrėžia galingą kalbą, kuri transformuoja (angl. *traversing*) kodą į sintaksės medžius ir gali išgauti informaciją iš šaltinio kodo. Internetiniame šaltinyje (Gra2Mol 2013) yra patalpinti įrankio išbandymo pavyzdžiai bei instrukcijos.

1.2.4.1.,„Gra2Mol“ aprašymas

Modelio išgavimas yra vykdomas iš pasenusios programinės įrangos kodo ir daugumos modernizuojamos programinės įrangos kodo scenarijų, aprašytų įvairiomis programavimo kalbomis. Įrankis, galintis programos kodą perteikti į KDM metamodelius, yra sukonstruotas taip, kad galėtų būti pritaikomas įvairioms programavimo kalboms.

6 pav. pateiktas įrankio vykdymo procesas.



6 pav. „Gra2Mol“ vykdomas procesas
(Šaltinis: Gra2Mol 2013)

Pagrindiniai transformacijos elementai yra:

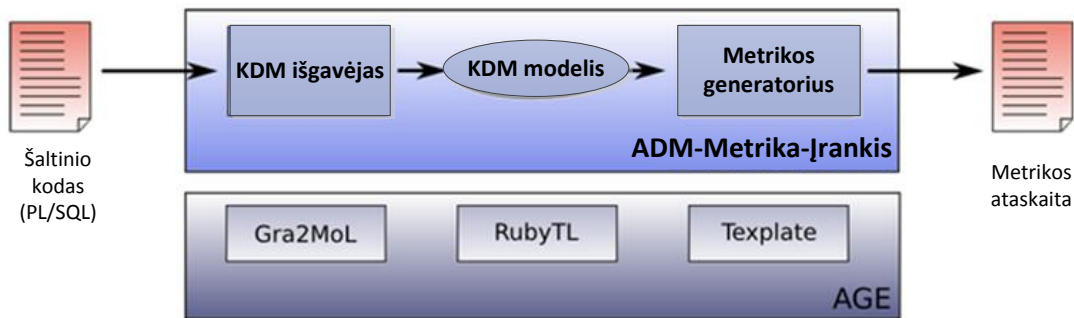
- Kodo gramatika (šaltinio kodas).
- „Gra2Mol“ vykdomas procesas.
- Tikslinis metamodelis.

„Gra2MoL“ vykdomas procesas prasideda nuo CST išgavimo. Šis įrankis iš kodo gramatikos sukuria medžio pavidalo sintaksę. „Gra2MoL“ variklis (angl. *engine*) yra parametrizuojamas: nurodomas „Gra2MoL“ transformacijos apibrėžimas (angl. *definition*), kuriame nurodytas metamodelis ir medžio pavidalo sintaksė. Rezultatas gaunamas iš „Gra2MoL“ įvykdyto proceso yra modelis, kuris atitinka varikliui nurodytą metamodelį.

1.2.4.2. „Gra2Mol“ pritaikymas

Pateiktame pavyzdyje (Gra2Mol) parodoma, kaip ADM (angl. *Architecture Driven Modernization*) gali būti praktiškai įgyvendinama per modernizavimo įrankį. Iliustruotos

dvi pagrindinės užduotys ADM procese: KDM metamodelio išgavimas ir šio modelio panaudojimas siekiant sugeneruoti modernizavimo procese dalyvaujančius artefaktus.



7 pav. „Gra2MoL“ modernizavimo įrankis
(Šaltinis: Izquierdo ir Molina 2010)

Pavyzdyje atvaizduotame 7 pav. naudojamas „Gra2MoL“ įrankis išgauti KDM modelius iš SQL „Oracle Forms Application“ procedūrinės kalbos kodo. Išgavimo procesas yra padalintas į du etapus: abstrakčios sintaksės medžio metamodelio (toliau – ASTM) gavimo iš kodo, ir šio tarpinio modelio transformacija į KDM modelius. „Gra2MoL“ yra naudojamas ASTM modeliui išgauti iš SQL procedūrinės kalbos kodo, ypač iš SQL trigerių. KDM modeliai buvo gauti iš „RubyTL“ apibrėžtų modelis į modelį transformacijų. ASTM ir KDM vienas kitą papildo modeliuojant programinės įrangos sintaksę ir semantiką. Panaudojant apibrėžtų metrikų generavimo įrankį „Texplate“ yra pateikiamas rezultatas – metrikos ataskaita.

1.2.4.3. Privalumai ir trūkumai

Įrankis pritaikomas trigerių nagrinėjimui. Pagrindinis privalumas, kad „Gra2MoL“ yra parametrizuojamas įrankis, todėl gali būti pritaikytas įvairioms programavimo kalboms. Tačiau iš to seka daugybė trūkumų:

- KDM turi labai didelį metamodelių pasirinkimą, bet pavyzdžių stoka daro metamodelį sunkiai suprantamu.
- Naudotojas turi dirbti tik su KDM susijusiais domenais norimos srities automatizavimui. Reikia pasirinkti tinkamą KDM metamodelį, kuris nustatytų egzistuojančios sistemos pagrindinius aspektus (domenus).

- ASTM sudarymas yra daug paprastesnis nei KDM, tačiau taip pat reikalingos medžio struktūros metamodelio sudarymo žinios.

1.2.5. SQL kalbos apdorojimo įrankis „General SQL parser“

„General SQL parser“ yra komercinis įrankis galintis apdoroti, formatuoti, modifikuoti ir analizuoti įvairias SQL kalbas. Internetiniame šaltinyje (General SQL parser 2013) pateikiamas platus aprašymas visų vykdomų funkcijų.

1.2.5.1. „General SQL parser“ aprašymas

„General SQL parser“ įrankis gali palengvinti darbą analizuojant SQL gramatiką. Įrankis yra palaikomas įvairiomis programavimo kalbomis: C#, VB.NET, Java, C/C++, Delphi, VB ir geba analizuoti duomenų bazių valdymo sistemų gramatikas: SQL Server, Oracle, DB2, MySQL, Teradata, PostgreSQL ir ACCESS.

Pagrindinės vykdomos funkcijos:

- Netvarkingo programavimo kodo sutvarkymas – daugelio SQL kalbų kodo atpažinimas ir tvarkingas atvaizdavimas.
- SQL sintaksės klaidų aptikimas – galimybė aptikti sintaksės klaidas neprisijungus prie duomenų bazės ir nevykdant SQL sakinio.
- Lentelių ir stulpelių pavadinimų išrinkimas – visų pavadinimų išrinkimas, atvaizdavimas bei šių duomenų apdorojimas.
- Sąlyginių SQL objektų išrinkimas (SELECT, INSERT, CREATE ir kita) – vykdomų procedūrų atpažinimas ir išrinkimas.
- Sąryšių išrinkimas DDL sakiniuose – lentelių sukūrimo sakinių analizė, lentelių pavadinimų išgavimas, atributų pavadinimai, duomenų tipai, pirminiai ar išoriniai raktai, unikalios reikšmės.
- SQL sakinių išskleistų medžio pavidalu pateikimas XML formatu – išskleidus XML formatu galimas rezultato tolimesnis panaudojimas analizuojant SQL gramatiką. Įrankis sukurtas taip, kad galima aprašyti sakinių išskleidimo standartus.

- Perrašymas ar konvertavimas SQL išraiškų į nurodytą kitą kalbą.
- Įvesties ir duomenų vientisumo nuostatis DB objektams – SQL kode galimybė rasti ir įvertinti objektus, su kuriais atliekami veiksmai (pvz., pateikiamas lentelės iškvietimų (angl. *select*) ar įrašymų (angl. *insert*) skaičius). Įrankis sukurtas taip, kad galima keisti fiksuojamų duomenų nuostačius.

1.2.5.2.Privalumai ir trūkumai

„General SQL parser“ įrankis yra pritaikomas trigerių nagrinėjimui, tačiau pagrindinis trūkumas, kad yra komercinės paskirties ir išgaunamos verslo žinios, o ne verslo taisyklės. Internetiniame šaltinyje pateikta įvairių demonstracinių naudojimo pavyzdžių. Galimi „Demo“ versijų parsisiuntimai pritaikyti kai kurioms programavimo kalbom (pvz., Java, C# ar kita). Reikalingos įvairių programavimo kalbų geros žinios.

2. SIŪLOMO METODO APRAŠYMAS

Šiame skyriuje pateikiamas siūlomo metodo aprašymas, detalizuoti reikalavimai prototipui. Reikalavimai siūlomam metodui ir jo prototipui yra suformuoti remiantis išnagrinėtais metodais, įrankiais ir atsižvelgiant į jų privalumus bei trūkumus. Skyriuje aprašyti verslo žodyno sudarymo metodai, kontrolės srautų grafų teorija bei pasiūlyti verslo taisyklių šablonai.

2.1. Metodo reikalavimai

Išnagrinėjus metodus ir įrankius, jų privalomus ir trūkumus, buvo suformuluoti šie reikalavimai.

Reikalavimai įėjai:

- Turi būti nagrinėjamas egzistuojančios MS SQL Server 2008 duomenų bazės valdymo sistemos DDL šaltinio kodas.
- Sistemų analitikas turi aprašyti verslo žodyno taisykles.
- Sistemų analitikas turi aprašyti verslo taisyklių šablonų taisykles.

Reikalavimai saugykloms:

- Verslo žodyno (terminų ir faktų) saugykla turi būti automatiškai papildoma.
- Verslo žodyno saugykloje turi būti vykdoma automatinė paieška ir egzistuojančių įrašų patikrinimas prieš įrašant naujus.
- Abstrakčios sintaksės medžio saugykla turi būti automatiškai papildoma.
- Kontrolės srauto grafų saugykla turi būti automatiškai papildoma.
- Šablonų saugykla turi būti papildoma sistemų analitiko.
- Šablonų saugykloje turi būti vykdoma paieška pagal šablonų taisykles.

Reikalavimai duomenų apdorojimui:

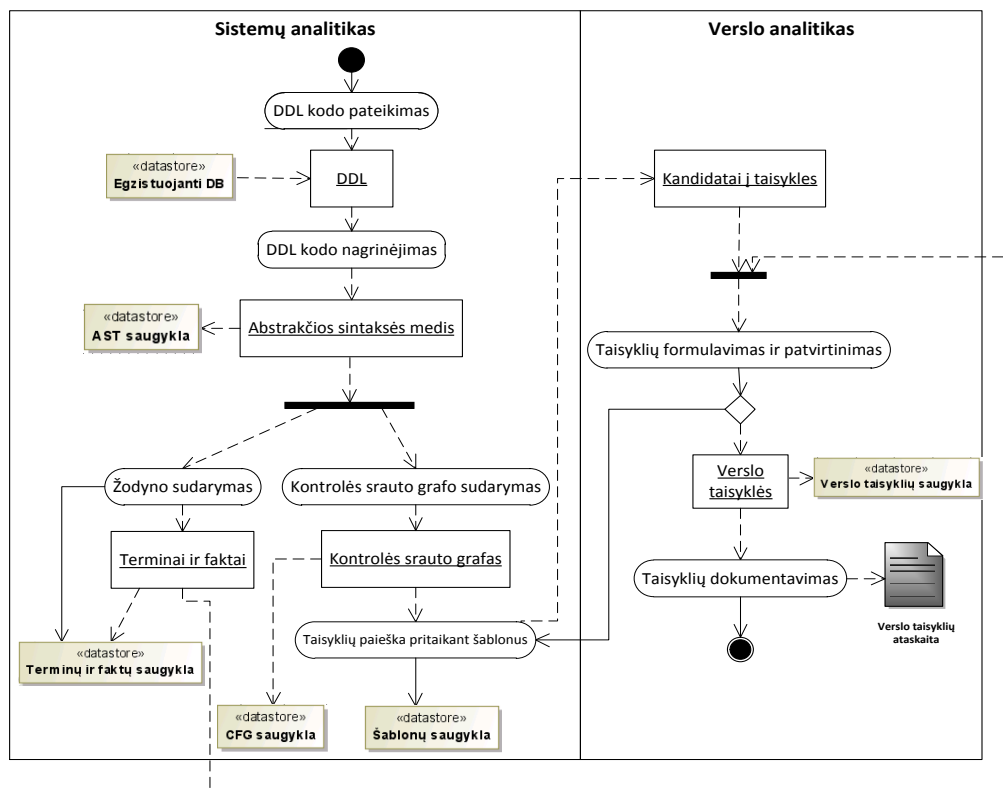
- Turi būti pusiau automatinis proceso vykdymas – įėjigos ir išeigos duomenų apdorojimui ar vykdomų taisyklių paruošimui dalyvauja sistemų analitikas ir ar verslo analitikas.
- DDL kodas turi būti išanalizuotas ir išsaugotas abstrakčios sintaksės medžio pavidalu.
- Sistemų analitikas turi galėti patvirtinti kandidatą į verslo žodyną arba jį atmesti.
- Abstrakčios sintaksės medis turi būti išanalizuotas ir jame rasti trigeriai išsaugoti kontrolės srauto grafų pavidalu.
- Iš abstrakčios sintaksės medžio turi būti išgautas verslo žodynas pagal aprašytas taisykles.
- Kontrolės srauto grafams turi būti taikomas paieškos gilyn algoritmas.
- Automatinis verslo taisyklės šablono parinkimas.
- Automatinis verslo žodyno pritaikymas verslo taisyklės šablonui.
- Verslo analitikas turi galėti patvirtinti sugeneruotą kandidatą į verslo taisyklę arba ją atmesti.

Reikalavimai išeigai:

- Sistemų analitikui turi būti pateikiami vertinimui kandidatai į verslo žodyną.
- Verslo analitikui turi būti pateikiami vertinimui kandidatai į verslo taisykles su verslo žodyno atitikmenimis.
- Turi būti, patvirtintų, saugykloje saugomų, verslo taisyklių sąrašo spausdinimas.

2.2. Verslo taisyklių išgavimo iš trigerių metodo aprašymas

Remiantis suformuluotais reikalavimais, magistro darbe yra siūlomas duomenų bazės trigeriuose realizuotų verslo taisyklių išgavimo metodas. 8 pav. pateikta metodo veiklos diagrama.



8 pav. Verslo taisyklių išgavimo iš trigerių veiklos diagrama

Sistemų analitikas nagrinėja egzistuojančios duomenų bazės DDL kodą ir yra atsakingas už šio kodo parengimą bei pateikimą įvairiais skirtingais architektūriniais programinės įrangos aspektais. Parengtas kodas yra analizuojamas ir išsaugomas abstrakčios sintaksės medžio pavidalu. Šis siūlomo metodo proceso etapas pasirinktas iš nagrinėtame šaltinyje (Normantas ir Vasilecas 2013) pateikto metodo. Tolimesniame etape vykdomos lygiagrečios veiklos: žodyno sudarymas bei kontrolės srauto grafo sudarymas.

Verslo žodyno terminai išgaunami iš duomenų bazės schemas. Taip pat, žodyno sudarymui iš AST medžio, sistemų analitikas aprašo trigeriuose vartojamų terminų ir faktų SQL sakinių tipų taisykles. Šiame proceso etape yra numatytas, išgautų terminų ir faktų, automatinis patikrinimas egzistuojančioje saugykloje – ar tokie įrašai jau yra saugomi. Patikrinti (nesami saugykloje) terminai ir faktai pateikiami sistemų analitikui tolimesniam jų vertinimui ir sprendimui, ką talpinti į saugyklą, o ko ne.

Kontrolės srauto grafo sudaromas iš AST medžio yra lygiagretus žodyno sudarymo veiklai. Tai yra dvi veiklos techniškai nepriklausančios viena nuo kitos. Pagal trigerių

raktinius žodžius, abstrakčios sintaksės medyje yra atpažįstami trigeriai ir jiems sukonstruojami kontrolės srauto grafai. Sukonstruoti kontrolės srauto grafai išsaugomi saugykloje.

Šablonų saugykloje yra saugomi verslo taisyklių šablonai. Kiekvienam šablonui yra aprašytos formalios taisyklės. Sistemų analitikas gali plėsti pritaikomų verslo taisyklių šablonų aibę.

Kiekvienam kontrolės srauto grafui yra pritaikomas paieškos gilyn algoritmas, įvertinamos visų kelių viršūnės ir vartojamų sakinių tipų raktiniai žodžiai. Įvertinus visas grafo viršūnes gali būti parenkamas atitinkamas verslo taisyklės šablonas išsaugotas saugykloje ir suformuojamas kandidatas į verslo taisyklę. Kandidatai pateikiami verslo analitikui tolimesniam apdorojimui.

Iš AST ir duomenų schemas išgavus verslo žodyną, atitinkami elementai yra įstatomi į verslo taisyklių kandidatus. Verslo analitikas peržiūri kandidatus į taisykles ir gali jas patvirtinti arba atmesti. Patvirtintos taisyklės yra išsaugomos verslo taisyklių saugykloje, o atmestos grąžinamos pakartotinam šablono pritaikymui (arba atmetimui).

Pasirinktas sąrašas išsaugotų verslo taisyklių gali būti atspausdinamas ataskaitoje.

2.3. Verslo žodyno sudarymas

Verslo taisyklės remiasi faktais, o faktai remiasi terminais, t. y., norint apibrėžti taisykles, reikia apibrėžti faktų tipus, o norint apibrėžti faktų tipus, reikia apibrėžti terminus.

Magistro darbe siūloma nagrinėti MS SQL Server 2008 duomenų bazėje realizuotus trigerius. Vartojant įvairius SQL sakinių tipus, trigerio kode, yra apibrėžiami papildomi žodyno elementai – terminai arba faktai:

- Terminai – tai aibė vardų suteiktų duomenų bazės duomenims: lentelių pavadinimai, atributų pavadinimai, sinonimai, kintamųjų deklaravimas, terminų apibrėžimai, kintamųjų užpildymas.

- Faktai – tai aibė terminų apjungimų ar išvedimų gaunamų vartojant įvairius trigerio kodo SQL sakinių tipus: kintamųjų reikšmių užpildymas, įvairūs išrinkimai.

Turint verslo žodyną yra sudaromas bazinių teiginių rinkinys – faktai – kas su kuo susiję, koks susiejimo tipas, kokie yra identifikuojantys atributai. Darbe siūloma apibrėžti pagrindinius kodo sakinių tipus, kur yra vartojami arba priskiriami nauji verslo terminai ar faktai:

- Kintamųjų deklaravimas (pvz., `DECLARE name varchar(10)`).
- Kintamųjų reikšmių užpildymas, kai reikšmės yra primityvūs tipai, terminų apibrėžimai (pvz., `SET @status = 'ok'`).
- Kintamųjų reikšmių užpildymas, kai reikšmės yra išvedamos iš kitų lentų, nauji terminai, faktai (pvz., `SET @status = (SELECT Name FROM Status)`).
- Sinonimai (SQL vadinami aliasais, pvz., `SELECT Name AS StudentName FROM Student`).
- Išrinkimai ir naudojamos sąlygos. Kiekvienas `WHERE` raktinio žodžio predikatas yra faktas. Prie tam tikrų faktų išvedamas naujas terminas ar terminų aibė (pvz., `SELECT name, surname FROM student WHERE student.male=female`).
- Išrinkimai ir naudojami apjungimai (angl. *join*) – kiekvienas apjungimas gali būti išvedamas į faktą.

Lentelėje 2 pateikiami kai kurie SQL sakinių tipai vartojami trigeriuose. Stulpelyje „Aprašymas“ pateikti MS SQL Server DBVS įvairūs SQL sakinių struktūros pavyzdžiai, kuriais remiantis galima aprašyti taisykles (Transact-SQL Keywords 2013; SQL condition 2013).

Priedo B paveiksle 14 pavaizduotas verslo žodyno elementų supaprastinto tipo metamodelis.

2 lentelė. SQL sakinių tipų šablonai terminų ir faktų išgavimui

Sakinio tipas	Aprašymas	Žodyno elementas
DECLARE	Naujų kintamųjų deklaravimas nurodant pavadinimą ir duomens tipą (naujo termino apibrėžimas) DECLARE @VariableName DataType	Terminas
SET	Kintamajam priskiriama reikšmė vartojant „=” operatorių (terminui priskiriama reikšmė) SET @VariableName = DesiredValue	Terminas
SET = (SELECT FROM)	Kintamajam priskiriama reikšmė vartojant operatorių „=” ir SELECT sakinį arba sakinius. Galimos grupės SELECT sakinių, kurios gražina reikšmę. Pirmenybė teikiama giliausiame gylyje esantiems skliaustams. Raktinis žodis FROM nurodo lentelę iš kurios imami duomenys SET @VariableName = (SELECT Name FROM Table)	Terminas
SELECT AS FROM	AS vartojamas SELECT sakinyje naujo lauko pavadinimui apibrėžti - sinonimas, aliasas. SELECT EmployeeNumber AS 'Number' FROM Employees	Terminas
SELECT FROM WHERE Sąlyga	Šis sakinio tipas vartojamas po sakinio tipo SET. Raktinis žodis WHERE vartojamas susiaurinti ar patikslinti rezultatų sąrašą. Sąlyga gali būti išreikšta įvairiais skaičiavimais vartojant lentelėje esančius duomenis SELECT column FROM table WHERE condition	Faktas
Sąlyga		
=	Eilučių radimas vartojant lygybės operatorių Name = 'term'	Faktas
LIKE	Eilučių radimas vartojant LIKE operatorių. Randamos visos reikšmės, kuriose yra nurodytas fragmentas	Faktas

Sakinio tipas		Aprašymas	Žodyno elementas																		
		<table><tr><td>Name</td><td>LIKE</td><td>(</td><td>'%Term%'</td><td>)</td></tr></table>	Name	LIKE	(	'%Term%'	)														
Name	LIKE	(	'%Term%'	)																	
<= < >= > !=		Eilučių radimas vartojant palyginimų operatorius <table><tr><td>Name</td><td><=</td><td>number</td></tr><tr><td></td><td><</td><td></td></tr><tr><td></td><td>>=</td><td></td></tr><tr><td></td><td>></td><td></td></tr><tr><td></td><td>!=</td><td></td></tr></table>	Name	<=	number		<			>=			>			!=		Faktas			
Name	<=	number																			
	<																				
	>=																				
	>																				
	!=																				
OR		Eilučių radimas, kurios gali tenkinti kelias sąlygas <table><tr><td></td><td>Name</td><td>=</td><td>'term'</td></tr><tr><td>OR</td><td>Name</td><td>=</td><td>number1</td></tr><tr><td>OR</td><td>Name</td><td>=</td><td>number2</td></tr></table>		Name	=	'term'	OR	Name	=	number1	OR	Name	=	number2	Faktas						
	Name	=	'term'																		
OR	Name	=	number1																		
OR	Name	=	number2																		
AND		Eilučių radimas, kurios turi tenkinti kelias sąlygas <table><tr><td></td><td>Name</td><td>LIKE</td><td>(</td><td>'%Term%'</td><td>)</td></tr><tr><td>AND</td><td>Name</td><td>LIKE</td><td>(</td><td>'Term%'</td><td>)</td></tr><tr><td>AND</td><td>Name</td><td>=</td><td></td><td>'Term'</td><td></td></tr></table>		Name	LIKE	(	'%Term%'	)	AND	Name	LIKE	(	'Term%'	)	AND	Name	=		'Term'		Faktas
	Name	LIKE	(	'%Term%'	)																
AND	Name	LIKE	(	'Term%'	)																
AND	Name	=		'Term'																	
IN		Eilučių radimas, kurios išvardytos sąraše <table><tr><td>Name</td><td>IN</td><td>(</td><td>'Term1'</td><td>,</td><td>'Term2'</td><td>,</td><td>'Term3'</td><td>)</td></tr></table>	Name	IN	(	'Term1'	,	'Term2'	,	'Term3'	)	Faktas									
Name	IN	(	'Term1'	,	'Term2'	,	'Term3'	)													
BETWEEN AND		Eilučių radimas, kurios turi reikmių tarp nurodytų rėžių <table><tr><td>Name</td><td>BETWEEN</td><td>Number1</td><td>AND</td><td>Number2</td></tr></table>	Name	BETWEEN	Number1	AND	Number2	Faktas													
Name	BETWEEN	Number1	AND	Number2																	

Sakinio tipas	Aprašymas	Žodyno elementas																																											
SELECT FROM Apjungimas ON	Duomenų apjungimas yra būdas išrinkti duomenis iš daugiau nei vienos lentelės vartojant visus stulpelius arba išrenkant tam tikras stulpelių reikšmes. Yra apibrėžiami trys etapai: - Lentelių apibrėžimas, kurios bus apjungtos. - Pasirenkamas stulpelis, kuris sukuria nuorodą į kiekvieną lentelę . - Parašomas SQL sakiny, kuris sukuria įrašus. <table><tr><td>SELECT</td><td>WhatColumn(s)</td><td>FROM</td><td>Table1</td><td>INNER JOIN</td><td>Table2</td><td>ON</td><td>Condition</td></tr><tr><td></td><td></td><td></td><td></td><td>CROSS JOIN</td><td></td><td>---</td><td>---</td></tr><tr><td></td><td></td><td></td><td></td><td>LEFT JOIN</td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td>RIGHT JOIN</td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td>FULL JOIN</td><td></td><td></td><td></td></tr></table> <i>WhatColumn(s)</i> – stulpelių sąrašas, kuriuos norima gauti apjungiant kelias lenteles. <i>Table1</i> yra lentelė, kurios duomenys bus priskiriami lentelei <i>Table2</i> pagal išorinį raktą (ChildrenTable). <i>Table2</i> yra lentelė, pagal kurio pirminio rakto stulpelį yra pasirenkami lentelės Table1 stulpelių įrašai (<i>Table2</i> stulpeliui priskiriama <i>Table1</i> stulpelio reikšmė) (ParentTable). <i>Condition</i> loginėje išraiškoje aprašomi tikrinami ar atrankami reikiami įrašai - stulpeliai <table><tr><td>Table1Column</td><td>=</td><td>Table2Column</td></tr></table>	SELECT	WhatColumn(s)	FROM	Table1	INNER JOIN	Table2	ON	Condition					CROSS JOIN		---	---					LEFT JOIN								RIGHT JOIN								FULL JOIN				Table1Column	=	Table2Column	Faktas
SELECT	WhatColumn(s)	FROM	Table1	INNER JOIN	Table2	ON	Condition																																						
				CROSS JOIN		---	---																																						
				LEFT JOIN																																									
				RIGHT JOIN																																									
				FULL JOIN																																									
Table1Column	=	Table2Column																																											
Apjungimas																																													
CROSS JOIN	Sudaromas sąrašas iš visų galimų stulpelių apjungimų. Pirmos lentelės pirmas stulpelis su antros lentelės visais stulpeliais iš eilės (neatrenkant); pirmos lentelės antras stulpelis su antros lentelės visais stulpeliais ir t.t. Dėl šios priežasties nereikia apibrėžti sąlygos su raktiniu žodžiu ON	Faktas																																											
INNER JOIN	INNER JOIN apjungimas naudojamas kai norima gauti sąrašą be tuščių reikšmių (NULL). Rezultatų sąrašė apjungiami visi įrašai iš <i>Table2</i> ir <i>Table1</i> , kur lentelėje Table1 išorinio rakto stulpelyje įrašai yra užpildyti	Faktas																																											

Sakinio tipas		Aprašymas	Žodyno elementas
		– ne NULL. Pvz., Select o.number, s.name from order o inner join status s on o.statusid = s.id -> inner join „griežta sąlyga“ – visi užsakymai ir būsenos)	
	LEFT JOIN	Apjungiant lenteles paimami visi <i>Table1</i> stulpeliai ir priskiriamos <i>Table2</i> lentelės reikšmės pagal <i>Table1</i> išorinį raktą įskaitant ir NULL stulpeliams (atitinkamai stulpelio eilutei priskiriama NULL reikšmė). Pvz., Select o.number, s.name from order o left join status s on o.statusid = s.id -> left join „lengva sąlyga“ – visi užsakymai ir būsenos jei yra	Faktas
	RIGHT JOIN	Apjungiant lenteles paimami visi <i>Table2</i> stulpeliai ir priskiriamos (išrenkamos atitinkamos) <i>Table1</i> lentelės reikšmės pagal <i>Table2</i> pirminį raktą. <i>Table2</i> nuskaitoma pirmoji eilutė ir atrankamos visos atitinkamos <i>Table2</i> reikšmės. Naujoje lentelėje sukurama tiek eilučių, kiek atitikmenų. Toliau seka atrinkimas pagal <i>Table2</i> antrą eilutę ir t.t.. <i>Table2</i> lentelėje nėra NULL reikšmių.	Faktas
	FULL JOIN	Apjungiant lenteles paimamos visos reikšmės iš <i>Table2</i> ir <i>Table1</i> lentelių. Jei įrašas iš vienos lentelės neturi atitinkamo įrašo kitoje lentelėje, tai reikšmė užpildoma NULL.	Faktas

2.4. Kontrolės srauto grafo sudarymas

Šiame skyriuje pateikiama šiek tiek teorijos apie formuojamus kontrolės srauto grafus pagal duomenų srautų analizės apibrėžimus (Khedker *et al.* 2009; Tip 1995) bei pagrindiniai taisyklių apibrėžimai, kuriais vadovaujantis siūloma formaliai aprašyti trigerio procedūrinio kodo sąlygas. Kai sąlygos tenkinamos, procedūrinis kodas yra „suprantamas“, kaip verslo taisyklės įgyvendinimas ir jam galima pritaikyti verslo taisyklės šablonus.

Apibrėžimas 1: Kontrolės srauto grafas (angl. *Control-flow graph*) yra visų kelių, kurie gali būti pereinami vykdant procedūrą, atvaizdavimas naudojant grafų notaciją.

Kontrolės srauto grafas žymimas $CFG=(V,E)$, kur CFG yra kryptinis grafas, V – viršūnių aibė (angl. *vertices*), E - briaunų aibė (angl. *edges*). Viršūnė $v_i \in V$ vaizduoja procedūros kodo bloką, o briauna $e_k \in E$ vaizduoja galimą kontrolės srautą $v_i \rightarrow v_j \in E$ iš viršūnės v_i į viršūnę v_j ($v_i, v_j \in V$). Sraute $v_i \rightarrow v_j \in E$, v_i – pirmtakas (angl. *predecessor*) v_j , o v_j – tęsėjas (angl. *successor*) v_i . Taip pat CFG turi specialias viršūnes *START* ir *STOP*, atitinkančias procedūros pradžią ir pabaigą.

Toliau grafo viršūnes vadinsime blokais. Blokas suprantamas kaip:

- Seka nuosekliai einančių kodo sakinių, turinčių vieną įėjimą ir vieną išėjimą tašką.
- Neturinčių jokių išsišakojimų (angl. *branches*) (pvz., IF, SWITCH, WHILE) tipo sakinių.
- Jei paprasti sakiniai (T-SQL UPDATE, INSERT, DELETE, SELECT, SET, DECLARE ar k.t.) gali būti apjungiami į vieną bloką (priklauso nuo poreikio analizuojamos informacijos grūdėtumui), tai išsišakojantys sakiniai (T-SQL IF, WHILE) yra visuomet atskiri blokai.

T-SQL palaikomi kontrolės srauto sakiniai pateikiami internetiniame šaltinyje (Control of Flow 2013).

Apibrėžimas 2: Blokas v_i esantis CFG vadinamas dominuojančiu (angl. *dominated*) prieš v_j ir žymimas $v_i \text{ dom } v_j$, jei visi keliai nuo įėjimo bloko *START* iki bloko v_j turi pereiti per bloką v_i . *START* blokas dominuoja prieš visus kitus blokus esančius CFG. $\text{Dom}(v_j)$ – aibė visų dominuojančių blokų prieš bloką v_j . Kiekvienas blokas dominuoja prieš save $v_j \in \text{Dom}(v_j)$.

Išsišakojančiuose sakiniuose (IF, WHILE) įėjimo blokas (IF sąlyga, WHILE sąlyga) dominuoja prieš visus kitus blokus esančius išsišakojimo sakinyje.

Apibrėžimas 3: Blokas v_i esantis CFG yra post-dominuojamas (angl. *post-dominated*) bloko v_j , jei visi keliai iš v_i iki *STOP* bloko pereina per bloką v_j . $\text{Pdom}(v_j)$ – aibė visų post-dominuojančių blokų blokui v_j . Kiekvienas blokas post-dominuoja sau $v_j \in \text{Pdom}(v_j)$.

Apibrėžimas 4: Blokas v_j turi kontrolės priklausomybę blokui v_i , jei egzistuoja kelias P nuo v_i iki v_j , toks kad tenkinamos sąlygos:

- Bet kuris $v_u \neq v_i, v_j$ esantis P yra post-dominuojamas v_j .
- Blokas v_i nėra post-dominuojamas bloko v_j .

Priedo B paveiksle 14 pavaizduotas kontrolės srauto grafo metamodelis.

2.5. Verslo taisyklių šablonų sudarymas

Verslo taisyklės šablonas susideda iš dviejų dalių: verslo taisyklės šablono pagal apibrėžtus taisyklių tipus ir trigerio įvykio šablono. Šablonams aprašomos formalios taisyklės, kurias tenkinant jie gali būti atrinkti ir pritaikomi kontrolės srauto grafams. Magistro darbe pateikiama aibė šablonų taisyklių tipų pritaikomų sudarytiems CFG.

2.5.1. Verslo taisyklės šablonų tipai

Lentelėje 3 pateikti verslo taisyklės šablono sudarymo pavyzdžiai atsižvelgiant į SQL kalboje vartojamus raktinius žodžius ir kontrolės srauto grafų taisyklių tipus, kuriais

vadovaujantis formaliai aprašytos trigerio procedūrinio kodo sąlygos. Stulpelyje „Pavadinimas“ pateikiama nuoroda į magistro prieduose patalpintus verslo taisyklių šablonų formalius aprašymus.

3 lentelė. Verslo taisyklių šablonų taisyklių tipų pavyzdžiai

Raktiniai žodžiai								
Taisyklių tipas	INSERT	UPDATE	DELETE	ROLLBACK	RAISERROR /PRINT	BRAKE	CONTINUE	PAVADINIM AS
IF (sąlyga)(1) (2)	(1)							T_1
	(1)				(2)			T_6
		(1)						T_2
		(1)			(2)			T_7
			(1)					T_3
			(1)		(2)			T_8
				(1)				T_4
				(1)	(1)			T_9
					(1)			T_5
WHILE (sąlyga) (1)	(1)							T_10
		(1)						T_11
			(1)					T_12
WHILE (sąlyga) (1)	(1)					(2)	(3)	T_13
IF (sąlyga)(2)		(1)				(2)	(3)	T_14
ELSE (3)			(1)			(2)	(3)	T_15

Remiantis šiais pavyzdžiais siūloma plėsti šablonų aibę. Skyriuje 2.6 paveiksle 14 pavaizduotas VT šablonų taisyklių tipų metamodelis.

2.5.1. Trigerio įvykio šablonų tipai

Atsižvelgiant į ECA taisykles, trigeris aktyvuojamas įvykus tam tikram įvykiui. Prie kiekvieno CFG yra saugoma tėvinė informacija, kokiam įvykiui įvykus aktyvuojamas trigeris. Priklausomai nuo įvykio, verslo taisyklėms išgauti turi būti taikomi praplėsti šablonai. Lentelėje 4 pateiktas raktinių žodžių kombinacijų detalizavimas ir jiems atitinkami šablonai.

4 lentelė. Trigerio įvykio raktinių žodžių aprašymai ir taikomi šablonai

Raktinių žodžių išraiška	Aprašymas	Šablonas
ON <terminas> AFTER/FOR INSERT	Trigeris aktyvuojamas prieš įterpiant įrašą į lentelę. Kai vykdomas AFTER INSERT trigeris, tuo pačiu metu sukuriamą laikina lentelė „inserted“ – joje laikoma sukurtų įrašų kopija.	ĮRAŠANT DUOMENIS <terminas>
ON <terminas> INSTEAD OF INSERT	Trigeris aktyvuojamas prieš įterpiant įrašą į lentelę. Skirtumas su AFTER trigeriu yra toks, kad galima atlikti apibrėžtus įvairius veiksmus dar prieš atliekant pakeitimus lentelėje ar rodinyje.	PRIEŠ ĮRAŠANT DUOMENIS <terminas>
ON <terminas> AFTER/FOR UPDATE	Trigeris aktyvuojamas prieš atnaujinant lentelės įrašą.	ATNAUJINANT DUOMENIS <terminas>
ON <terminas> INSTEAD OF UPDATE	Trigeris aktyvuojamas prieš atnaujinant lentelės įrašą. Skirtumas su AFTER trigeriu yra toks, kad galima atlikti apibrėžtus įvairius veiksmus dar prieš atliekant pakeitimus lentelėje ar rodinyje.	PRIEŠ ATNAUJINANT DUOMENIS <terminas>
ON <terminas> AFTER/FOR DELETE	Trigeris aktyvuojamas prieš lentelės įrašo ištrynimą. Kai vykdomas AFTER DELETE trigeris, tuo pačiu metu sukuriamą laikina lentelė „deleted“ – joje laikoma sukurtų įrašų kopija.	IŠTRINANT DUOMENIS <terminas>
ON <terminas> INSTEAD OF DELETE	Trigeris aktyvuojamas prieš lentelės įrašo ištrynimą. Skirtumas su AFTER trigeriu yra toks, kad galima atlikti apibrėžtus įvairius veiksmus dar prieš atliekant pakeitimus lentelėje ar rodinyje.	PRIEŠ IŠTRINANT DUOMENIS <terminas>

Trigerius aktyvuojantys įvykiai gali būti jungtiniai – susidedantys iš kelių arba apjungiantys visus tris galimus įvykių raktinius žodžius. Lentelėje 5 pateiktas jungtinių raktinių žodžių kombinacijų detalizavimas ir atitinkamas šablonas.

5 lentelė. Trigerių įvykių jungtinės raktinių žodžių išraiškos ir šablonai

Raktinių žodžių išraiška	Šablonas
--------------------------	----------

Raktinių žodžių išraiška	Šablonas
ON <terminas> AFTER/FOR INSERT UPDATE	ĮRAŠANT ARBA ATNAUJINANT DUOMENIS <terminas>
ON <terminas> AFTER/FOR INSERT DELETE	ĮRAŠANT ARBA IŠTRINANT DUOMENIS <terminas>
ON <terminas> AFTER/FOR UPDATE DELETE	ATNAUJINANT ARBA IŠTRINANT DUOMENIS <terminas>
ON <terminas> AFTER/FOR INSERT UPDATE DELETE	ĮRAŠANT, ATNAUJINANT ARBA IŠTRINANT DUOMENIS <terminas>
ON <terminas> INSTEAD OF INSERT UPDATE	PRIEŠ ĮRAŠANT ARBA ATNAUJINANT DUOMENIS <terminas>
ON <terminas> INSTEAD OF INSERT DELETE	PRIEŠ ĮRAŠANT ARBA IŠTRINANT DUOMENIS <terminas>
ON <terminas> INSTEAD OF UPDATE DELETE	PRIEŠ ATNAUJINANT ARBA IŠTRINANT DUOMENIS <terminas>
ON <terminas> INSTEAD OF INSERT UPDATE DELETE	PRIEŠ ĮRAŠANT, ATNAUJINANT ARBA IŠTRINANT DUOMENIS <terminas>

2.6. Verslo taisyklės šablono pritaikymo algoritmas

Priede B pateikti AST, CFG, verslo žodyno ir verslo taisyklių šablonų taisyklių tipų metamodelių susiejimai. Remiantis supaprastintais metamodeliais, magistro darbe pateikiamas, pseudokalba parašytas, algoritmas, skirtas verslo taisyklės šablono parinkimui ir pritaikymui konkrečiam kontrolės srauto grafo taisyklių tipui. Algoritmas pateiktas priede C.

Algoritmo detalizavimas:

- Įeiga – algoritmui paduodami duomenys:
 - Visiems triggeriams suformuotų CFG aibė.
 - CFG kiekvieno kelio aplangytų blokų aibė (randama pritaikius paieškos gilyn algoritmą).
 - Šablonų aibė.
 - Šablonų taisyklių tipų aibė.

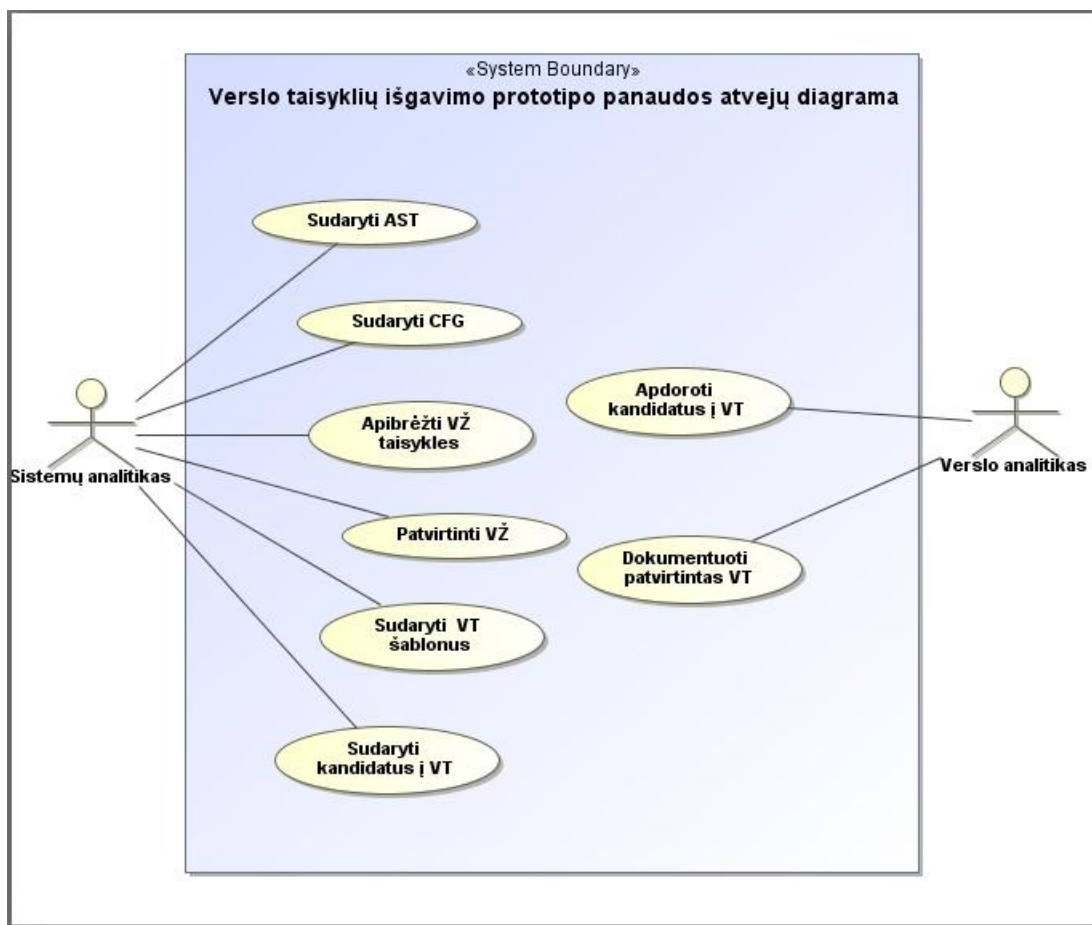
- Tikrinamas CFG kiekvieno kelio blokas:
 - Pradedama nuo START bloko.
 - Tikrinamas iš eilės einantis TRUE ar FALSE kelias į kitą bloką.
 - Patikrinama ar VT šablonų saugykloje egzistuoja formalus taisyklės aprašymas, tenkinantis praeitą bloką (pvz., IF šablonų taisyklių tipų aibė).
 - Jei TAIP, ir jei tai paskutinis STOP blokas, tai iškviečiama funkcija, kuri pritaiko šabloną pagal kelio nustatytą taisyklių tipą.
 - Jei TAIP, ir jei tai ne paskutinis STOP blokas, tęsiamas tikrinimas iš eilės einančio kito bloko.
 - Jei NE, nutraukiamas algoritmo vykdymas.
- Verslo taisyklės šablonui pritaikomas įvykio šablonas.
- Išėiga – algoritmo rezultatas yra aibė:
 - Kandidatai į verslo taisykles (verslo žodynas neužpildytas).

3. PRAKTINĖ DALIS

Remiantis metodo reikalavimais, šiame skyriuje, pateikta prototipo funkcionalumo specifikacija atvaizduota panaudos atvejų diagrama, veikėjų (sistemos ir verslo analitikų) požiūriu. Suformuota ir aprašyta prototipo architektūra. Architektūroje naudojami moduliai detalizuoti kiekvienas atskirai ir pateikiamas nuoseklus pavyzdys, apimantis pilną prototipo funkcionalumo įgyvendinimą. Metodo veiksmingumas patikrintas rankiniu būdu ir aprašytas eksperimentiniame metodo įvertinime.

3.1. Metodą palaikančios priemonės prototipo funkcionalumo specifikacija

Magistro darbe pateikiama panaudos atvejų diagrama (angl. *use-case*), kurioje pavaizduota priemonės elgsena sistemos analitiko bei verslo analitiko požiūriu. 9 pav. pavaizduotos penkios sistemos analitiko ir dvi verslo analitiko inicijuojamos užduotys bei lentelėse detalizuotos užduotys.



9 pav. Verslo taisyklių išgavimo iš pasenusios DBVS prototipo panaudos atvejų diagrama

Užduočių detalizavimas:

6 lentelė. Sudaryti AST užduoties detalizavimas

Aprašymas	Sudaryti AST	
Prieš sąlyga	Sistemų analitikas prisijungia prie sistemos	
Vykdymas	Žingsnis	Veiksmas
	1.	DDL kodo įkėlimas
	2.	AST metamodelio aprašymas
	3.	AST generavimas
Po sąlyga	Abstrakčios sintaksės medis išsaugotas AST saugykloje	
Išimtys	Žingsnis	Veiksmas
	1.	Įkeltas ne DDL kodas
	2.	Nepavyksta įrašyti duomenų

7 lentelė. Sudaryti CFG užduoties detalizavimas

Aprašymas	Sudaryti CFG	
Prieš sąlyga	Sudarytas AST metamodelis	
Vykdymas	Žingsnis	Veiksmas
	1.	AST metamodelio įkėlimas
	2.	CFG metamodelio aprašymas
	3.	CFG generavimas
Po sąlyga	Kontrolės srauto grafas išsaugotas CFG saugykloje	
Išimtys	Žingsnis	Veiksmas
	1.	Įkeltas ne AST metamodelis
	2.	Nepavyksta įrašyti duomenų

8 lentelė. Apibrėžti verslo žodyno taisykles užduoties detalizavimas

Aprašymas	Apibrėžti verslo žodyno taisykles	
Prieš sąlyga	Sistemų analitikas inicijuoja verslo žodyno taisyklių apibrėžimą	
Vykdymas	Žingsnis	Veiksmas
	1.	Apibrėžti terminų taisykles
	2.	Apibrėžti faktų taisykles
	3.	Išgauti verslo žodyną pagal apibrėžtas taisykles
	4.	Patikrinti išgautų VŽ elementų dublikatus saugykloje
	5.	Pateikti patikrintus VŽ lementus
Po sąlyga	Sistemų analitikui pateikiama kandidatų į verslo žodyno elementų aibė	
Išimtys	Žingsnis	Veiksmas
	1.	Visi išgauti VŽ elementai jau egzistuoja saugykloje
	2.	Sistemų analitikui pateikiama tuščia aibė VŽ elementų

9 lentelė. Patvirtinti verslo žodyną užduoties detalizavimas

Aprašymas	Patvirtinti verslo žodyną	
Prieš sąlyga	Sistemų analitikui pateikiama kandidatų į verslo žodyno elementų aibė	
Vykdymas	Žingsnis	Veiksmas
	1.	Patvirtinti VŽ elementą
	2.	Atmesti VŽ elementą
	3.	Išsaugoti patvirtintus VŽ lementus VŽ saugykloje
Po sąlyga	VŽ saugykloje išsaugoti patvirtinti terminai ir faktai	
Išimtys	Žingsnis	Veiksmas
	1.	Visi kandidatai į VŽ elementus atmetami
	2.	Nepavyksta įrašyti duomenų

10 lentelė. Sudaryti verslo žodyno šablonus užduoties detalizavimas

Aprašymas	Sudaryti verslo žodyno šablonus	
Prieš sąlyga	Sistemų analitikas prisijungia prie sistemos	
Vykdymas	Žingsnis	Veiksmas
	1.	Apibrėžti VT šablonų taisyklių tipus
	2.	Apibrėžti taisyklių tipams verslo taisyklių šablonus
	3.	Išsaugoti verslo taisyklių šablonus VT saugykloje
Po sąlyga	Sistemų analitikas atsijungia nuo sistemos	
Išimtys	Žingsnis	Veiksmas
	1.	Taisyklių tipui neapibrėžtas VT šablonas
	2.	Nepavyksta įrašyti duomenų

11 lentelė. Sudaryti kandidatus į verslo taisykles užduoties detalizavimas

Aprašymas	Sudaryti kandidatus į verslo taisykles	
Prieš sąlyga	Sudaryti CFG išsaugoti CFG saugykloje	
Vykdymas	Žingsnis	Veiksmas
	1.	Pateikti CFG
	2.	Vykdyti paieškos gilyn algoritmą
	3.	CFG keliui rasti atitinkamą VT taisyklių tipą
	4.	Pritaikyti verslo taisyklės šabloną
	5.	Pritaikyti įvykio šabloną
	6.	Pateikti kandidatą į verslo taisyklę
Po sąlyga	Kandidatai į verslo taisykles pateikti verslo analitikui	
Išimtys	Žingsnis	Veiksmas
	1.	CFG nerastas kelias, kuriam būtų pritaikomas VT taisyklių tipas

12 lentelė. Apdoroti kandidatus į verslo taisykles užduoties detalizavimas

Aprašymas	Apdoroti kandidatus į verslo taisykles	
Prieš sąlyga	Verslo analitikas prisijungia prie sistemos	
Vykdymas	Žingsnis	Veiksmas
	1.	Pateikti kandidatus į verslo taisykles
	2.	Pritaikyti verslo žodyno lementus
	3.	Patvirtinti verslo taisyklę
	4.	Atmesti verslo taisyklę
	5.	Išsaugoti patvirtintą verslo taisyklę
Po sąlyga	Verslo analitikas išsaugo patvirtintas verslo taisykles VT saugykloje	
Išimtys	Žingsnis	Veiksmas
	1.	Nepateikta kandidatų į verslo taisykles
	2.	Nerasti tinkami verslo žodyno lementai

	3.	Atmesti visi kandidatai į verslo taisykles
	4.	Nepavyksta įrašyti duomenų

13 lentelė. Dokumentuoti patvirtintas verslo taisykles

Aprašymas	Dokumentuoti patvirtintas verslo taisykles	
Prieš sąlyga	Verslo analitikas prisijungia prie sistemos	
Vykdymas	Žingsnis	Veiksmas
	1.	Atlikti verslo taisyklių paiešką
	2.	Spausdinti verslo taisyklių sąrašą
Po sąlyga	Atspausdinamas word tipo ataskaitos dokumentas	
Išimtys	Žingsnis	Veiksmas
	1.	Nerasta nei viena verslo taisyklė
	2.	Nerastas spausdintuvas

3.2. Prototipo architektūra

Realizuojamos priemonės architektūroje svarbios dvi programinės įrangos platformos: „General SQL Parser“ ir „Eclipse“, kuriose yra įgyvendinti aštuoni skirtingų modulių funkcionalumai. Realizuojamos priemonės prototipe pavaizduotos šešios duomenų saugyklos. Paveiksle 10 pateikta verslo taisyklių išgavimo iš pasenusios DBVS prototipo architektūra.

„General SQL Parser“ platformoje realizuojami moduliai:

- Kodo nagrafinėjimo modulis.
- Žodyno išgavimo modulis.
- Žodyno tvirtinimo modulis.

„Eclipse“ platformoje realizuojami moduliai:

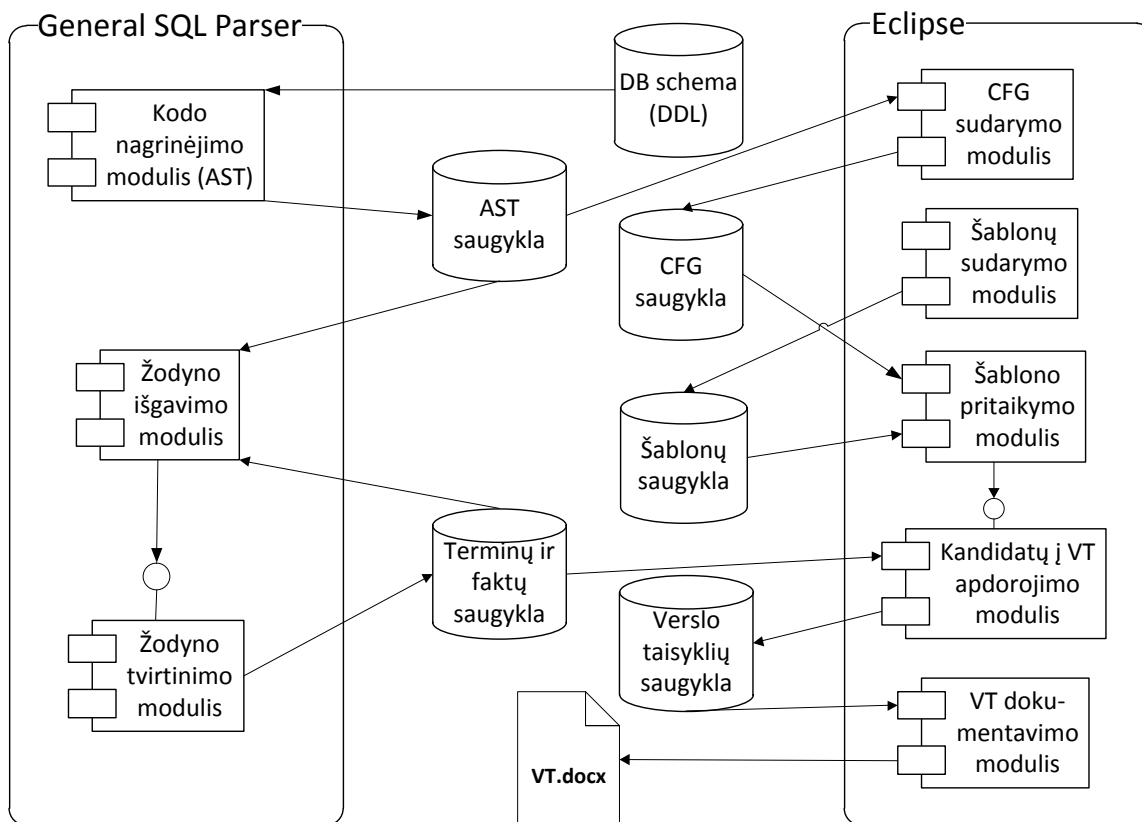
- CFG sudarymo modulis.
- Šablonų sudarymo modulis.
- Šablonų pritaikymo modulis.
- Kandidatų į verslo taisykles modulis.

- Verslo taisyklių dokumentavimo modulis.

Kiekviename komponente yra apdorojami įeigos duomenys ir pateikiami išeigos duomenys tolimesnėms veikloms. Visi duomenys saugomi saugyklose:

- DB schema (DDL) – pagrindinis nagrinėjamas šaltinis – DDL kodas aprašantis duomenų struktūrą.
- AST saugykla – saugomas sugeneruotas abstrakčios sintaksės medis.
- CFG saugykla – saugomi sugeneruoti trigerių kontrolės srauto grafai.
- Šablonų saugykla – saugomi sistemų analitiko aprašyti verslo taisyklių šablonai.
- Terminų ir faktų saugykla – saugomi išgauti verslo žinių duomenys.
- Verslo taisyklių saugykla – saugomos verslo analitiko patvirtintos taisyklės.

Realizuojamos priemonės prototipo rezultatas yra spausdinamas word tipo ataskaitos dokumentas.



10 pav. Verslo taisyklių išgavimo iš pasenusios DBVS prototipo architektūra

Modulių funkcionalumo įgyvendinimui aktyviai dalyvauja žmogiškasis faktorius: sistemų analitikas aprašo įvairias taisykles bei sistemų ir verslo analitikai apdoroja gautus tarpinius rezultatus.

3.2.1. Modulių įgyvendinimas

Šiame skyriuje detalizuoti prototipo modulių veikimai ir pateikiamas verslo taisyklių išgavimo iš egzistuojančios MS SQL Server 2008 duomenų bazės trigerių pavyzdys.

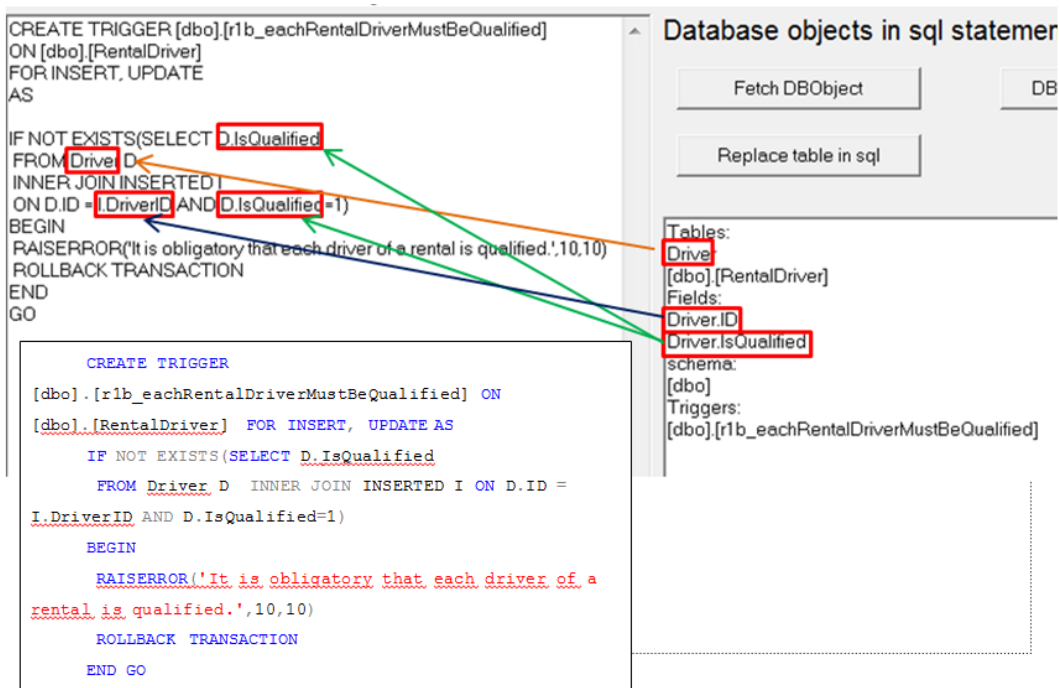
3.2.1.1. Kodo nagrinėjimo modulis

Iš egzistuojančios MS SQL Server 2008 duomenų bazės saugyklos yra paduodamas DDL kodas ir sugeneruojamas abstrakčios sintaksės medis. AST išsaugomas

saugykloje XML formatu ir pateikiamas sistemų analitikui. Tokiu būdu apdorotas DDL kodas naudojamas semantinei analizei.

3.2.1.2. Žodyno išgavimo modulis

Panaudojant „General SQL Parser“ įrankį iš triggerio kodo galima išgauti verslo žodyno elementus: lentelių pavadinimai bei jų atributai. Paveiksle 11 pavaizduotas gautas rezultatas panaudojant įrankį.



11 pav. Žodyno elementų išgavimas naudojant „General SQL Parser“ įrankį

Šiuo būdu galima išgauti visus verslo žodyno elementus vartojamus triggeriuose, tačiau tai nebūtų detali duomenų informacija ir nebūtų pakankamai naudinga analitikui.

Šis magistro darbas neapėmė praplėsto verslo žodyno išgavimo tyrimo. Tačiau, kad pasiekti tikslingesnių rezultatų, yra siūloma, sistemų analitikui, aprašyti taisykles ir išgauti praplėstą verslo žodyną (trigeryje vartojamus terminus ir faktus) iš AST (žiūrėti skyriuje 2.3). Išgavus verslo žodyno elementus, prieš pateikiant sistemų analitikui, patikrinama ar tokie įrašai dar neegzistuoja saugykloje. Įrašomi tik nauji verslo žodyno elementai.

3.2.1.3. Žodyno tvirtinimo modulis

Remiantis prielaida, kad yra vykdomas praplėstas verslo žodyno išgavimo modulis, žodyno tvirtinimo modulyje, sistemų analitikui yra pateikiami, saugykloje patikrinti, kandidatai į verslo žodyno lementus (terminai ir faktai). Sistemų analitikas gali patvirtinti kandidatą ir išsaugoti verslo žodyno saugykloje arba atmesti jį kaip netinkamą.

3.2.1.4. Kontrolės srauto grafo sudarymo modulis

Siūlomame metode, naudojant „Eclipse“ įrankį, yra suformuojami kontrolės srauto grafai pagal duomenų srautų analizės apibrėžimus (Khedker *et al.* 2009; Tip 1995).

Žemiau pateiktas rankiniu būdu sudarytas kontrolės srauto grafo pavyzdys. Siūloma tokiu principu magistro darbe sudaryti CFG trigeriams ir išsaugoti juos CFG saugykloje.

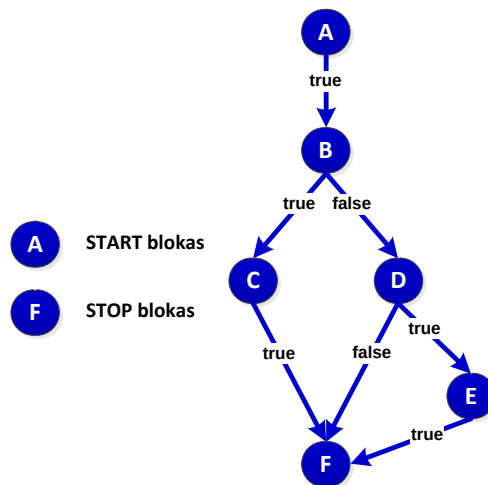
Pavyzdys:

Pateiktame paveiksle 12 MS SQL Server 2008 trigerio kodas yra išskirstytas blokais (A...F):

```
CREATE TRIGGER ria_Demo
ON DiscountValues
FOR INSERT, UPDATE
AS
  A
  DECLARE @MaxVal INT = (SELECT MinValue FROM AllowedValues)
  DECLARE @MinVal INT = (SELECT MaxValue FROM AllowedValues)
  DECLARE @CurVal INT = (SELECT Value FROM INSERTED)
  B
  IF (@CurVal >= @MaxVal)
  C
  BEGIN
    RAISERROR('The value should not be more than allowed value.', 10, 10)
    ROLLBACK TRANSACTION
  END
  ELSE
  D
  IF (@CurVal < @MinVal * 0.10)
  E
  BEGIN
    RAISERROR('The value should not be less than 10% of minimum allowed value.', 10, 10)
    ROLLBACK TRANSACTION
  END
  F
```

12 pav. MS SQL Server 2008 trigerio kodo išskirstymas blokais

Kontrolės srauto grafas:



13 pav. Trigerio kodo atvaizdavimas kontrolės srauto grafu

3.2.1.5. Šablono sudarymo modulis

Šiame modulyje, „Eclipse“ įrankiu, sistemų analitikas aprašo ir šablonų saugykloje išsaugo paruoštus verslo taisyklių šablonus. Šablonų saugykla gali būti papildoma.

Pavyzdys (tęsinys):

Lentelėje 14 pateiktas VT šablono taisyklės pavyzdys atitinkantis pavyzdyje nagrinėjamą trigerio kodą ir jam sudarytą CFG.

14 lentelė. Taisyklė, aprašanti verslo taisyklės šabloną

Pavadinimas	Verslo taisyklė <u>draudžianti</u> duomenų modifikavimą, kai prieštaraujama tam tikroms dviem sąlygoms.
Apibūdinimas	Trigerio kūne turi būti if_statement sakiny, kuris savo arba then_body , turi turėti rollback_transaction sakinį. Ir else_body bloke turi būti if_statement sakiny, kuris savo arba then_body turi turėti rollback_transaction sakinį T.y. prie tam tikrų dviejų tikrinamų sąlygų, trigeris nutraukia duomenų modifikavimo transakciją.

Pavadinimas	Verslo taisyklė <u>draudžianti</u> duomenų modifikavimą, kai prieštaraujama tam tikroms dviem sąlygoms.
Taisyklės formalus aprašymas	Trigerio procedūros kontrolės srauto grafe CFG turi būti du keliai $P_1 = \{v_i, \dots, v_n\}$, $P_2 = \{v_i, \dots, v_n\}$, kuriuose būtų blokai $v_{rollback_1} \in P_1$, $v_{rollback_2} \in P_2$ apimantys ROLLBACK TRANSACTION sakinius bei turintys kontrolės priklausomybę blokams $v_{if_1} \in P_1$, $v_{if_2} \in P_2$ apimantiems IF...[ELSE]...sakinius, kad būtų tenkinama sąlyga: $v_{if_1} \in Dom(v_{rollback_1}) \wedge v_{rollback_1} \in Pdom(v_{if_1}) \wedge v_{if_1} \in Dom(v_{if_2}) \wedge v_{if_2} \in Pdom(v_{if_1}) \wedge v_{if_2} \in Dom(v_{rollback_2}) \wedge v_{rollback_2} \in Pdom(v_{if_2})$

3.2.1.6. Šablono pritaikymo modulis

Šablonų pritaikymo modulyje kontrolės srauto grafo keliui yra vykdomas verslo taisyklės šablono pritaikymo algoritmas (žiūrėti skyriuje 2.6) ir įvertinus, kokie blokai ir kokie sakinių tipai yra grafe pritaikomas atitinkamas verslo taisyklės šablonas. Kandidatų į verslo taisyklės sudarymas vykdomas automatiniu būdu pagal įvertintus kontrolės srautų grafus ir šablonų taisyklių formalius aprašymus – parenkamas verslo taisyklės šablonas. Įstatomos reikšmės ir gautas rezultatas – kandidatas į verslo taisyklę – yra pateikiamas verslo analitikui.

Pavyzdys (tęsinys):

Lentelėje 15 pateiktas, pavyzdyje nagrinėjamam trigeriui, pritaikytas verslo taisyklės šablonas.

15 lentelė. Šablonų pritaikymas

Pavadinimas	Verslo taisyklė <u>draudžianti</u> duomenų modifikavimą, kai prieštaraujama tam tikroms dviem sąlygoms.
Programų sistemų lygmens taisyklės šablonas	JEI TENKINAMA <sąlyga>, TAI ATŠAUKTI DUOMENŲ MODIFIKAVIMĄ IR PARODYTI PRANEŠIMĄ <„pranešimas“> IR JEI TENKINAMA <sąlyga>, TAI ATŠAUKTI DUOMENŲ MODIFIKAVIMĄ IR PARODYTI PRANEŠIMĄ <„pranešimas“>

Pavadinimas	Verslo taisyklė <u>draudžianti</u> duomenų modifikavimą, kai prieštaraujama tam tikroms dviem sąlygoms.
Verslo taisyklės šablonas	NEGALIMA ATNAUJINTI REIKŠMIŲ IR PARODOMAS PRANEŠIMAS KAI NETENKINAMOS SĄLYGOS: 1. <sąlyga> - <„pranešimas“>; 2. <sąlyga> - <„pranešimas“>.

Patikrinus trigerio įvykį, ir pritaikius šabloną (FOR INSERT UPDETA raktinių žodžių kombinacijai (žiūrėti lentelėje 5)), yra praplečiama anksčiau pavyzdyje suformuota verslo taisyklė (lentelė 16).

16 lentelė. Šablonų pritaikymas atsižvelgiant į įvykį

Įvykio šablonas	Verslo taisyklės šablonas
ĮRAŠANT ARBA ATNAUJINANT DUOMENIS <terminas>	NEGALIMA ATNAUJINTI REIKŠMIŲ IR PARODOMAS PRANEŠIMAS KAI NETENKINAMOS SĄLYGOS: 1. <sąlyga> - <„pranešimas“>; 2. <sąlyga> - <„pranešimas“>.

3.2.1.7. Kandidatų į verslo taisyklės apdorojimo modulis

Remiantis prielaida, kad yra vykdomas praplėstas verslo žodyno išgavimo modulis, šiame kandidatų į verslo taisyklės apdorojimo modulyje siūloma vykdyti automatinį terminų ir faktų „įstatymą“ į verslo taisyklių šablonus. Sugeneruoti kandidatai į verslo taisyklės yra pateikiami verslo analitikui tolimesniam apdorojimui. Verslo analitikas gali patvirtinti kandidatą ir išsaugoti verslo taisyklių saugykloje arba atmesti kaip netinkamą.

Pavyzdys (tęsinys):

Įstačius į šabloną reikšmes gaunamas galutinis kandidatas į verslo taisyklę ir tokiu formatu pateikiamas verslo analitikui (žiūrėti 17 lentelė).

17 lentelė. Verslo analitikui pateikiamas kandidatas į verslo taisyklę

Kandidatas į įvykio šabloną	Kandidatas į verslo taisyklę
------------------------------------	-------------------------------------

Kandidatas į įvykio šabloną	Kandidatas į verslo taisyklę
ĮRAŠANT ARBA ATNAUJINANT DUOMENIS <DiscountValues> ¹	NEGALIMA ATNAUJINTI REIKŠMIŲ IR PARODOMAS PRANEŠIMAS KAI NETENKINAMOS SĄLYGOS: 1. <@CurVal>=@MaxVal> ² - <,,The value should not be more than allowed value“>; 2. <@CurVal<@MinVal*0.10> ³ - <,, The value should not be less than 10 % of minimum allowed value“>.

3.2.1.8. Verslo taisyklių dokumentavimo modulis

Išsaugotas saugykloje, patvirtintas verslo taisykles, verslo analitikas gali peržiūrėti ir atsispausdinti. Galimas pilnas sąrašo spausdinimas ataskaitoje arba atlikus filtravimą pagal norimus kriterijus.

¹ Pavyzdyje nedetalizuotas verslo žodyno elementų įstatymas.

² Pavyzdyje nedetalizuotas verslo žodyno elementų įstatymas.

³ Pavyzdyje nedetalizuotas verslo žodyno elementų įstatymas.

3.3. Eksperimentinis metodo įvertinimas

Šiame skyriuje aprašomas eksperimentas metodo įgyvendinimui ir tinkamumui patikrinti. Eksperimentui buvo naudojama magistro darbe suformuota verslo taisyklių šablonų aibė (žiūrėti Priedas A), egzistuojančios MS SQL Server 2008 duomenų bazės DDL kodas su 22 realizuotais trigeriais. Eksperimentinį tyrimą, rankiniu būdu, atliko 4 asmenys: 2 sistemų analitikai (duomenų bazių ekspertai, dirbantys atitinkamai 7 ir 10 metų), 2 verslo analitikai (pasirinkti 2 verslo informacinių sistemų analitikai, dirbantys atitinkamai 2 ir 12 metų).

Rankiniu būdu atlikus visus metode aprašytus etapus buvo gauti tokie rezultatai (18 lentelė).

18 lentelė. Eksperimentinio tyrimo rezultatai

Analizuojami elementai: 22 trigeriai	Kontrolės srauto grafo taisyklių tipas			
	IF..	WHILE..	WHILE..IF..ELSE..	Kita
1-mas sistemų analitikas ir 1-mas verslo analitikas				
Suformuota CFG tipų	9	2	3	8
CFG atitinka šablonų taisyklių tipus	7	1	1	-
Pritaikyta VT šablonų	7	1	1	-
Pritaikyta įvykio šablonų	7	1	1	-
Patvirtinti kandidatai į VT	6	1	1	-
2-tas sistemų analitikas ir 2-tas verslo analitikas				
Suformuota CFG tipų	9	3	3	7
CFG atitinka šablonų taisyklių tipus	7	2	1	-
Pritaikyta VT šablonų	7	2	1	-
Pritaikyta įvykio šablonų	7	2	1	-
Patvirtinti kandidatai į VT	6	2	1	-

Lentelės 18 duomenys rodo, kad:

- Priklausomai nuo sistemų analitiko kompetencijos, buvo gauti skirtingi grupių rezultatai. Reikalingos geros kontrolės srautų grafų sudarymo žinios, kad tinkamai atlikti esminį metodo etapą.
- Įvykdžius verslo taisyklės šablono algoritmą, rasta aibė CFG kelių taisyklių tipų, tenkinančių formaliai aprašytus šablonų taisyklių tipus. Rezultatai skiriasi dėl pirmame etape gauto skirtingo kiekio suformuotų CFG.
- Visiems CFG keliams, kurie tenkino formaliai aprašytus šablonų taisyklių tipus, buvo pritaikytas verslo taisyklės šablonas.
- Visiems CFG keliams, kuriems pritaikytas verslo taisyklės šablonas, buvo pritaikytas ir įvykio šablonas.
- Ne visi suformuoti kandidatai į verslo taisykles buvo patvirtinti. Rezultatai skiriasi dėl pirmame etape gauto skirtingo kiekio suformuotų CFG.
- Priklausomai nuo verslo analitiko kompetencijos, buvo gauti skirtingi grupių rezultatai. Reikalingos geras verslo veiklos ir verslo žodyno žinojimas norint priimti teisingus sprendimus.
- Suformuotų CFG aibė parodė, kad praktikoje naudojamų trigerių įvairovė yra didesnė, nei magistro darbe pasiūlytų analizuojamų CFG taisyklių tipų. Dėl to VT išgavimo efektyvumas ženkliai sumažėjo – apytiksliai 36 % trigerių buvo neištirti.

Atliktas tyrimas parodė, kad rankinių būdų įvykdžius visus siūlomo metodo etapus, lyginant rezultatus su pasiūlyta verslo taisyklių šablonų aibe, gautas rezultatas yra teigiamas ir patvirtina metodo efektyvumą.

REZULTATAI IR IŠVADOS

1. Išnagrinėjus taisyklių įgyvendinimo duomenų bazių valdymo sistemose principus, pastebėta, kad tikslių duomenų bazėse realizuojamų verslo taisyklių apibrėžtų susitarimų nėra ir dažniausiai reliacinėse DB jos įgyvendinamos trigeriais. Pasenusios informacinių sistemų programos grindžiamos tiksliais verslo taisyklėmis, kurios sudaro organizacijos veiklos pagrindą, bet šių sistemų išlaikymas yra labai brangus ir sudėtingas arba kartais net neįgyvendinamas. Tačiau kai atsiranda daug taisyklių, jų palaikymas tampa vis sudėtingesnis. Iškyla taisyklių galutinio užbaigimo, rekursijos fiksavimo, sudėtingumo nustatymo ir daugelis kitų problemų.
2. Išanalizavus įgyvendintus žinių išgavimo metodus bei įrankius, išsiaiškinta, jog dauguma metodų yra skirti kodo analizei, žodyno įrašų išgavimui, metamodelių sudarymui. Tačiau nėra nė vieno metodo, nagrinėjančio duomenų bazėse realizuotų trigerių išgaunant iš jų verslo taisykles.
3. Įvertinus išanalizuotų metodų privalomumus ir trūkumus, buvo suformuluoti reikalavimai naujam taisyklių išgavimui iš duomenų bazių trigerių metodui, taip pat reikalavimai duomenų įėjai, saugykloms, duomenų apdorojimui bei išeigai. Pasiūlytas metodas yra pusiau automatinis ir vykdomas dviejų rolių: sistemų ir verslo analitiko.
4. Remiantis suformuluotais reikalavimais, pasiūlytas taisyklių išgavimo iš duomenų bazių trigerių metodas bei jį palaikančios priemonės prototipo architektūra. Architektūra susideda iš šešių saugyklų: egzistuojančios DB, AST, terminų ir faktų, CFG, šablonų ir verslo taisyklių; aštuonių modulių: kodo nagrinėjimo, žodyno išgavimo, žodyno tvirtinimo, CFG sudarymo, šablonų sudarymo, šablonų pritaikymo, kandidatų į verslo taisykles apdorojimo bei verslo taisyklių dokumentavimo. Moduliai realizuojami naudojant „General SQL Parser“, „Eclipse“ įrankius ir įgyvendina visas metodo veiklas.
5. Metodo tinkamumui įvertinti buvo suformuoti 15 verslo taisyklių šablonų ir juos aprašančių taisyklių tipų. Rankiniu būdu atlikti tyrimai parodė, kad,

įvykdžius visus siūlomo metodo etapus, lyginant rezultatus su pasiūlyta verslo taisyklių šablonų aibe, gautas rezultatas yra teigiamas ir patvirtina metodo efektyvumą.

6. Siūloma plėsti verslo taisyklių šablonų aibę, nes atliktas tyrimas taip pat parodė, kad trigerių kodas yra labai įvairus (tai priklauso nuo programuotojo žinių ar nusistovėjusios jo praktikos aprašyti kodą) ir apytiksliai 36 % iš tyrimui pateiktų trigerių buvo neištirti.

LITERATŪRA

Adreescu, A. I.; Uta, A.I. 2007. Business Rules – Important Artefacts of Software Development, in *Economy Informatics*, 11-14 [žiūrėta 2013 m. gegužės 26 d.]. Prieiga per internetą: <<http://www.economyinformatics.ase.ro/content/EN7/Uta.pdf>>.

Avdejenkov, V. 2009. *Verslo taisyklių valdymo sistemų taikymo įmonėse tyrimas, Daktaro disertacija*. VGTU leidykla TECHNIKA [žiūrėta 2013 m. gegužės 26 d.]. Prieiga per internetą: <http://vddb.library.lt/fedora/get/LT-eLABa-0001:E.02~2009~D_20090504_091600-38718/DS.005.0.01.ETD>.

Badia, A. 2006. Business Rules in Databases, in *Encyclopedia of Database Technologies and Applications*. United States of America: Idea Group Reference, 47-53.

Baronas, R. 2002. *Duomenų bazių sistemos*. Vilnius: TEV. 82 p.

Create trigger [interaktyvus] 2013. [žiūrėta 2013 m. gegužės 26 d.]]. Prieiga per internetą: <<http://msdn.microsoft.com/en-us/library/ms189799.aspx>>.

Control of Flow [interaktyvus] 2013. [žiūrėta 2013 m. gegužės 26 d.]. Prieiga per internetą: <<http://msdn.microsoft.com/en-us/library/ms174290.aspx>>.

Dadashzadeh, M. 2006. Set Comparison in Relational Query Language, in *Encyclopedia of Database Technologies and Applications*. United States of America: Idea Group Reference, 624-631.

Gra2Mol [interaktyvus] 2013. [žiūrėta 2013 m. gegužės 26 d.]. Prieiga per internetą: <<http://code.google.com/a/eclipselabs.org/p/gra2mol/>>.

Izquierdo, J. L. C.; Molina, J. G. 2010. An Architecture-Driven Modernization Tool for Calculating Metrics, in *by the IEEE*.

Khedker, U.; Sanyal, A.; Karkare, B. 2009. Data Flow Analysis: Theory and Practice, in *CRC Press, Inc.*. Boca Raton, FL, USA [žiūrėta 2013 m. gegužės 26 d.]. Prieiga per internetą: <<http://www.cse.iitb.ac.in/~uday/dfaBook-web/>>.

Kolber, A.; Hay, D.; Healy, K.A.; Hall, J. 1997. GUIDE Business Rules Project, Final Reportr. 1.2, in *GUIDE International Corporation*.

Laučius, J.; Vasilecas, O. 2008. *Informacinių sistemų projektų ir kokybės valdymas: mokomoji knyga*, 2-oji laida. Vilnius: Technika. 78 p.

Leintz, B. P.; Swanson, E. F. 1980. Software maintenance Management, in *Addison-Wesley Longman Publishing Co., Inc.* Boston, MA.

Nelson, M. L. 1996. A Survey of Reverse Engineering and Program Comprehension, in *ODU CS 551 – Software Engineering Survey*.

Normantas, K.; Vasilecas, O. 2013. Extracting Business Vocabulary from Existing Databases.

Object Management Group 2002. *Business Rules in Models, Request For Information* [žiūrėta 2013 m. gegužės 26 d.]. Prieiga per internetą: <<http://www.omg.org/cgi-bin/doc?ad/02-09-13>>.

Object Management Group 2011. *Knowledge Discovery Metamodel Specification Version 1.3* [žiūrėta 2013 m. gegužės 26 d.]. Prieiga per internetą: <<http://www.omg.org/spec/KDM/1.3/PDF/>>.

Object Management Group 2008. *Semantics of Business Vocabulary And Business Rules v1.0* [žiūrėta 2013 m. gegužės 26 d.]. Prieiga per internetą: <<http://www.omg.org/spec/SBVR/1.0/>>.

Paradauskas, B.; Laurikaitis, A. 2006. Business knowledge extraction from legacy information systems, in *Information Technology and Control*, Vol. 35, No.3.

Paton, N. V. 1999. *Active rules in database systems*. New York: Springer.

Rodrigues, N. F.; Barbosa, L. S. 2010. Slicing for architectural analysis, in *Sci. Comput. Program.* 75, Issue 10 [žiūrėta 2013 m. gegužės 26 d.]. Prieiga per internetą: <<http://repositorium.sdum.uminho.pt/bitstream/1822/12250/1/RB10-preprint.pdf>> .

Ross, R. 2003. Principles of the Business Rules Approach, in *Addison-Wesley*.

SQL condition [interaktyvus] 2013. [žiūrėta 2013 m. gegužės 26 d.]. Prieiga per internetą: <<http://msdn.microsoft.com/en-us/library/ms188047.aspx>>.

Tip, F. 1995. A Survey of Program Slicing Techniques, in *Computer Science/Department of Software Technology*, 121-189.

Transact-SQL Keywords [interaktyvus] 2013. [žiūrėta 2013 m. gegužės 26 d.]. Prieiga per internetą: <<http://functionx.com/sqlserver/>>.

Transakcija [interaktyvus] 2013. [žiūrėta 2013 m. gegužės 26 d.]. Prieiga per internetą: <<http://lt.wikipedia.org/wiki/Transakcija>>.

Vasilecas, O.; Normantas, K. 2011. Deriving Business Rules from the models of existing information systems, in *International Conference on Computer Systems and Technologies – CompSysTech'11*.

Wang, X.; Sun, J.; Yang, X. 2004. Business Rules Extraction from Large Legacy Systems, in *Computer Society*,

Weiser, M. 1981. Program slicing, in *ICSE '81 Proceedings of the 5th international conference on Software Engineering*. Piscataway, NJ, USA [žiūrėta 2013 m. gegužės 26 d.]. Prieiga per internetą: <<http://dl.acm.org/citation.cfm?id=802557>> .

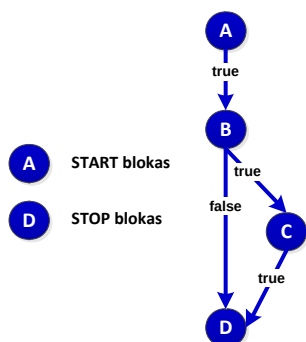
PRIEDAI

A priedas.

Verslo taisyklių šablonai

Šablonų aibė: (IF..) taisyklių tipas

Kontrolės srauto grafas:



Duomenų modifikavimas

VT šablonų pavyzdžiai tenkinantys CFG taisyklių tipą:

19 lentelė. Kontrolės srauto grafo (IF..) verslo taisyklės šablonas T_1

Pavadinimas	Verslo taisyklė <u>įrašanti</u> duomenis, jei duomenų modifikavimas tenkina tam tikrą sąlygą.
Apibūdinimas	Trigerio kūne turi būti if_statement sakiny, kuris savo arba then_body bloke turi turėti insert sakinį. T.y. prie tam tikros sąlygos, trigeris įrašo apibrėžtus duomenis.
Taisyklės formalus aprašymas	Trigerio procedūros kontrolės srauto grafe <i>CFG</i> turi būti bent vienas kelias $P = \{v_i, \dots, v_n\}$, kuriame būtų blokas $v_{insert} \in P$, apimantis INSERT.. [SET] .. [FROM] .. [WHERE] .. sakinį bei turintis kontrolės priklausomybę blokui $v_{if} \in P$, apimančiam IF... [ELSE] ... sakinį, kad būtų tenkinama sąlyga: $v_{if} \in Dom(v_{insert}) \wedge v_{insert} \in Pdom(v_{if})$
Programų sistemų lygmens taisyklės šablonas	JEI TENKINAMA <sąlyga>, ĮTERPTI DUOMENIS, KUR <faktas>
Verslo taisyklės šablonas	DUOMENŲ <faktas> ĮRAŠYMAS GALIMAS TIK JEI <sąlyga>

Pavadinimas	Verslo taisyklė <u>įrašanti</u> duomenis, jei duomenų modifikavimas tenkina tam tikrą sąlygą.
Pavyzdys	<pre> CREATE TRIGGER product_quantity_check ON products AFTER INSERT,UPDATE AS BEGIN SET NOCOUNT ON; declare @quantity int = null; SET @quantity = (select QuantityInStock from inserted); IF (@quantity <= 0) BEGIN INSERT INTO [MokymaiDB].[dbo].[Messages] ([MessageType] , [MessageText]) VALUES (1 , 'Sandėlyje nėra prekių. Reikia užsakyti prekių') END END </pre>

20 lentelė. Kontrolės srauto grafo (IF..) verslo taisyklės šablonas T_2

Pavadinimas	Verslo taisyklė <u>atnaujinanti</u> duomenis, jei duomenų modifikavimas tenkina tam tikrą sąlygą.
Apibūdinimas	Trigerio kūne turi būti if_statement sakiny, kuris savo arba then_body bloke turi turėti update sakinį. T.y. prie tam tikros sąlygos, trigeris atnaujina apibrėžtus duomenis.
Taisyklės formalus aprašymas	Trigerio procedūros kontrolės srauto grafe CFG turi būti bent vienas kelias $P = \{v_i, \dots, v_n\}$, kuriame būtų blokas $v_{update} \in P$, apimantis UPDATE..[SET]..[FROM]..[WHERE].. sakinį bei turintis kontrolės priklausomybę blokui $v_{if} \in P$, apimančiam IF...[ELSE]... sakinį, kad būtų tenkinama sąlyga: $v_{if} \in Dom(v_{update}) \wedge v_{update} \in Pdom(v_{if})$
Programų sistemų lygmens taisyklės šablonas	JEI TENKINAMA <sąlyga>, ATNAUJINTI DUOMENIS, KUR <faktas>
Verslo taisyklės šablonas	DUOMENŲ <faktas> ATNAUJINIMAS GALIMAS TIK JEI <sąlyga>
Pavyzdys	<pre> CREATE TRIGGER product_quantity_update ON OrderItems AFTER INSERT,UPDATE AS BEGIN SET NOCOUNT ON; declare @quantity int = null; SET @quantity = (select Quantity from inserted); IF (@quantity > 0) BEGIN UPDATE [MokymaiDB].[dbo].[Products] SET [QuantityInStock] = [QuantityInStock] - @quantity WHERE ID = (select ProductID from inserted) END END </pre>

21 lentelė. Kontrolės srauto grafo (IF..) verslo taisyklės šablonas T_3

Pavadinimas	Verslo taisyklė <u>ištrinantį</u> duomenis, jei duomenų modifikavimas tenkina tam tikrą sąlygą.
Apibūdinimas	Trigerio kūne turi būti if_statement sakiny, kuris savo arba then_body bloke turi turėti delete sakinį. T.y. prie tam tikros sąlygos, trigeris atnaujiną apibrėžtus duomenis.
Taisyklės formalus aprašymas	Trigerio procedūros kontrolės srauto grafe <i>CFG</i> turi būti bent vienas kelias $P = \{v_i, \dots, v_n\}$, kuriame būtų blokas $v_{delete} \in P$, apimantis DELETE...[SET]...[FROM]...[WHERE]... sakinį bei turintis kontrolės priklausomybę blokui $v_{if} \in P$, apimančiam IF...[ELSE]... sakinį, kad būtų tenkinama sąlyga: $v_{if} \in Dom(v_{delete}) \wedge v_{delete} \in Pdom(v_{if})$
Programų sistemų lygmens taisyklės šablonas	JEI TENKINAMA <sąlyga>, IŠTRINTI DUOMENIS, KUR <faktas>
Verslo taisyklės šablonas	DUOMENŲ <faktas> ŠALINIMAS GALIMAS TIK JEI <sąlyga>
Pavyzdys	<pre> CREATE TRIGGER [dbo].[product_settings_orders_update3] ON [dbo].[ProductSettings] AFTER INSERT, UPDATE AS BEGIN SET NOCOUNT ON; declare @actionID int = null; SET @actionID = (select ActionID from inserted); IF (@actionID = 5) BEGIN DELETE FROM Orders WHERE OrderState = 0 END END END </pre>

22 lentelė. Kontrolės srauto grafo (IF..) verslo taisyklės šablonas T_4

Pavadinimas	Verslo taisyklė <u>draudžianti</u> duomenų modifikavimą prieštaraujantį tam tikrai sąlygai.
Apibūdinimas	Trigerio kūne turi būti if_statement sakiny, kuris savo arba then_body bloke turi turėti rollback_transaction sakinį. T.y. prie tam tikros sąlygos, trigeris nutraukia duomenų modifikavimo transakciją.
Taisyklės formalus aprašymas	Trigerio procedūros kontrolės srauto grafe <i>CFG</i> turi būti bent vienas kelias $P = \{v_i, \dots, v_n\}$, kuriame būtų blokas $v_{rollback} \in P$, apimantis ROLLBACK TRANSACTION sakinį bei turintis kontrolės priklausomybę blokui $v_{if} \in P$, apimančiam IF...[ELSE]... sakinį, kad būtų tenkinama sąlyga: $v_{if} \in Dom(v_{rollback}) \wedge v_{rollback} \in Pdom(v_{if})$

Pavadinimas	Verslo taisyklė <u>draudžianti</u> duomenų modifikavimą prieštaraujantį tam tikrai sąlygai.
Programų sistemų lygmens taisyklės šablonas	JEI TENKINAMA <sąlyga>, NUTRAUKTI VEIKSMĄ
Verslo taisyklės šablonas	DUOMENŲ KOREGAVIMAS NEGALIMAS JEI <sąlyga>
Pavyzdys	<pre> CREATE TRIGGER [dbo].[product_quantity_update2] ON [dbo].[OrderItems] AFTER INSERT,UPDATE AS BEGIN SET NOCOUNT ON; declare @quantity int = null; SET @quantity = (select Quantity from inserted); IF (@quantity < 0) BEGIN ROLLBACK TRANSACTION END END END </pre>

Duomenų modifikavimas ir pranešimo rodymas

VT šablonų pavyzdžiai tenkinantys CFG taisyklių tipą:

23 lentelė. Kontrolės srauto grafo (IF..) verslo taisyklės šablonas T_5

Pavadinimas	Verslo taisyklė <u>parodanti pranešimą</u> , jei duomenų modifikavimas tenkina tam tikrą sąlygą.
Apibūdinimas	Trigerio kūne turi būti if_statement sakiny, kuris savo arba then_body bloke turi turėti print/raiserror sakinį. T.y. prie tam tikros sąlygos, trigeris parodo pranešimą.
Taisyklės formalus aprašymas	Trigerio procedūros kontrolės srauto grafe CFG turi būti bent vienas kelias $P = \{v_i, \dots, v_n\}$, kuriame būtų blokas $v_{print/raiserror} \in P$, apimantis PRINT/RAISERROR sakinį bei turintis kontrolės priklausomybę blokui $v_{if} \in P$, apimančiam IF...[ELSE]... sakinį, kad būtų tenkinama sąlyga: $v_{if} \in Dom(v_{print/raiserror}) \wedge v_{print/raiserror} \in Pdom(v_{if})$
Programų sistemų lygmens taisyklės šablonas	JEI TENKINAMA <sąlyga>, PARODYTI PRANEŠIMĄ <„pranešimas“>
Verslo taisyklės šablonas	PARODOMAS PRANEŠIMAS: <„pranešimas“> JEI <sąlyga>
Pavyzdys	<pre> CREATE TRIGGER [dbo].[product_quantity_update3] ON [dbo].[OrderItems] AFTER INSERT,UPDATE AS BEGIN SET NOCOUNT ON; declare @quantity int = null; </pre>

Pavadinimas	Verslo taisyklė <u>parodanti pranešimą</u> , jei duomenų modifikavimas tenkina tam tikrą sąlygą.
	<pre> SET @quantity = (select Quantity from inserted); IF (@quantity <= 0) BEGIN RAISERROR ('Klaida. Kiekis turi būti > 0.', 16, 1); END ELSE PRINT N'Prekių kiekis > 0'; END </pre>

24 lentelė. Kontrolės srauto grafo (IF..) verslo taisyklės šablonas T_6

Pavadinimas	Verslo taisyklė <u>parodanti pranešimą</u> ir <u>įrašanti</u> duomenis, jei duomenų modifikavimas tenkina tam tikrą sąlygą.
Apibūdinimas	Trigerio kūne turi būti if_statement sakiny, kuris savo arba then_body bloke turi turėti insert ir turi turėti print/raiserror sakinius. T.y. prie tam tikros sąlygos, trigeris įrašo apibrėžtus duomenis ir parodo pranešimą.
Taisyklės formalus aprašymas	Trigerio procedūros kontrolės srauto grafe CFG turi būti bent vienas kelias $P = \{v_i, \dots, v_n\}$, kuriame būtų blokas $v_{insert} \in P$, apimantis INSERT...[SET]...[FROM]...[WHERE]... sakinį, blokas $v_{print/raiserror} \in P$, apimantis PRINT/RAISERROR sakinį, ir būtų kontrolės priklausomybė blokui $v_{if} \in P$, apimančiam IF...[ELSE]... sakinį, kad būtų tenkinama sąlyga: $v_{if} \in Dom(v_{insert}) \wedge v_{insert} \in Pdom(v_{if}) \wedge$ $v_{if} \in Dom(v_{print/raiserror}) \wedge v_{print/raiserror} \in Pdom(v_{if}) \wedge v_{insert} \in$ $Dom(v_{print/raiserror}) \wedge v_{print/raiserror} \in Pdom(v_{insert})$
Programų sistemų lygmens taisyklės šablonas	JEI TENKINAMA <sąlyga>, ĮTERPTI DUOMENIS, KUR <faktas> IR PARODYTI PRANEŠIMĄ <„pranešimas“>
Verslo taisyklės šablonas	DUOMENŲ <faktas> ĮRAŠYMAS GALIMAS TIK JEI <sąlyga> IR PARODOMAS PRANEŠIMAS: <„pranešimas“>
Pavyzdys	<pre> CREATE TRIGGER [dbo].[product_quantity_check2] ON [dbo].[Products] AFTER INSERT,UPDATE AS BEGIN SET NOCOUNT ON; declare @quantity int = null; SET @quantity = (select QuantityInStock from inserted); IF (@quantity <= 0) BEGIN INSERT INTO [MokymaiDB].[dbo].[Messages] ([MessageType] , [MessageText]) VALUES (1 , 'Sandėlyje nėra prekių. Reikia užsakyti prekių') PRINT N'Sandėlyje nėra prekių. Reikia užsakyti prekių'; END END </pre>

25 lentelė. Kontrolės srauto grafo (IF..) verslo taisyklės šablonas T_7

Pavadinimas	Verslo taisyklė <u>parodanti pranešimą</u> ir <u>atnaujinanti</u> duomenis, jei duomenų modifikavimas tenkina tam tikrą sąlygą.
Apibūdinimas	Trigerio kūne turi būti if_statement sakiny, kuris savo arba then_body bloke turi turėti update ir turi turėti print/raiserror sakinius. T.y. prie tam tikros sąlygos, trigeris atnaujiną apibrėžtus duomenis ir parodo pranešimą.
Taisyklės formalus aprašymas	Trigerio procedūros kontrolės srauto grafe CFG turi būti bent vienas kelias $P = \{v_i, \dots, v_n\}$, kuriame būtų blokas $v_{update} \in P$, apimantis UPDATE...[SET]...[FROM]...[WHERE]... sakinį, blokas $v_{print/raiserror} \in P$, apimantis PRINT/RAISERROR sakinį, ir būtų kontrolės priklausomybė blokui $v_{if} \in P$, apimančiam IF...[ELSE]... sakinį, kad būtų tenkinama sąlyga: $v_{if} \in Dom(v_{update}) \wedge v_{update} \in Pdom(v_{if}) \wedge$ $v_{if} \in Dom(v_{print/raiserror}) \wedge v_{print/raiserror} \in Pdom(v_{if}) \wedge v_{update} \in$ $Dom(v_{print/raiserror}) \wedge v_{print/raiserror} \in Pdom(v_{update})$
Programų sistemų lygmens taisyklės šablonas	JEI TENKINAMA <sąlyga>, ATNAUJINTI DUOMENIS, KUR <faktas> IR PARODYTI PRANEŠIMĄ <„pranešimas“>
Verslo taisyklės šablonas	DUOMENŲ <faktas> ATNAUJINIMAS GALIMAS TIK JEI <sąlyga> IR PARODOMAS PRANEŠIMAS: <„pranešimas“>
Pavyzdys	<pre> CREATE TRIGGER product_quantity_update2 ON OrderItems AFTER INSERT,UPDATE AS BEGIN SET NOCOUNT ON; declare @quantity int = null; SET @quantity = (select Quantity from inserted); IF (@quantity > 0) BEGIN UPDATE [MokymaiDB].[dbo].[Products] SET [QuantityInStock] = [QuantityInStock] - @quantity WHERE ID = (select ProductID from inserted) PRINT N'Duomenys atnaujinti'; END END </pre>

26 lentelė. Kontrolės srauto grafo (IF..) verslo taisyklės šablonas T_8

Pavadinimas	Verslo taisyklė <u>parodanti pranešimą</u> ir <u>ištrinantį</u> duomenis, jei duomenų modifikavimas tenkina tam tikrą sąlygą.
Apibūdinimas	Trigerio kūne turi būti if_statement sakiny, kuris savo arba then_body bloke turi turėti delete ir turi turėti print/raiserror sakinius. T.y. prie tam tikros sąlygos, trigeris ištrina apibrėžtus duomenis ir parodo pranešimą.

Pavadinimas	Verslo taisyklė <u>parodanti pranešimą</u> ir <u>ištrinti</u> duomenis, jei duomenų modifikavimas tenkina tam tikrą sąlygą.
Taisyklės formalus aprašymas	Trigerio procedūros kontrolės srauto grafe CFG turi būti bent vienas kelias $P = \{v_i, \dots, v_n\}$, kuriame būtų blokas $v_{delete} \in P$, apimantis DELETE...[SET]...[FROM]...[WHERE]... sakinį, blokas $v_{print/raiserror} \in P$, apimantis PRINT/RAISERROR sakinį, ir būtų kontrolės priklausomybė blokui $v_{if} \in P$, apimančiam IF...[ELSE]... sakinį, kad būtų tenkinama sąlyga: $v_{if} \in Dom(v_{delete}) \wedge v_{delete} \in Pdom(v_{if}) \wedge$ $v_{if} \in Dom(v_{print/raiserror}) \wedge v_{print/raiserror} \in Pdom(v_{if}) \wedge v_{delete} \in$ $Dom(v_{print/raiserror}) \wedge v_{print/raiserror} \in Pdom(v_{delete})$
Programų sistemų lygmens taisyklės šablonas	JEI TENKINAMA <sąlyga>, IŠTRINTI DUOMENIS, KUR <faktas> IR PARODYTI PRANEŠIMĄ <„pranešimas“>
Verslo taisyklės šablonas	DUOMENŲ <faktas> ŠALINIMAS GALIMAS TIK JEI <sąlyga> IR PARODOMAS PRANEŠIMAS: <„pranešimas“>
Pavyzdys	<pre>CREATE TRIGGER [dbo].[product_settings_orders_update3] ON [dbo].[ProductSettings] AFTER INSERT,UPDATE AS BEGIN SET NOCOUNT ON; declare @actionID int = null; SET @actionID = (select ActionID from inserted); IF (@actionID = 5) BEGIN DELETE FROM Orders WHERE OrderState = 0 PRINT N'Duomenys istrinti; END END</pre>

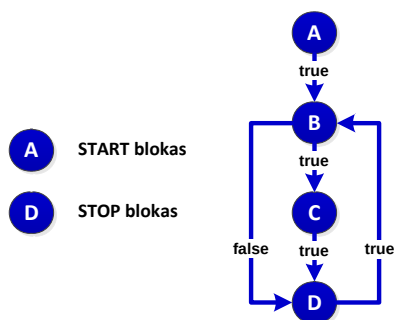
27 lentelė. Kontrolės srauto grafo (IF..) verslo taisyklės šablonas T_9

Pavadinimas	Verslo taisyklė <u>parodanti pranešimą</u> ir <u>draudžianti</u> duomenų modifikavimą prieštaraujantį tam tikrai sąlygai.
Apibūdinimas	Trigerio kūne turi būti if_statement sakiny, kuris savo arba then_body bloke turi turėti rollback_transaction ir turi turėti print/raiserror sakinius. T.y. prie tam tikros sąlygos, trigeris nutraukia duomenų modifikavimo transakciją ir parodo pranešimą.
Taisyklės formalus aprašymas	Trigerio procedūros kontrolės srauto grafe CFG turi būti bent vienas kelias $P = \{v_i, \dots, v_n\}$, kuriame būtų blokas $v_{rollback} \in P$, apimantis ROLLBACK TRANSACTION sakinį, blokas $v_{print/raiserror} \in P$, apimantis PRINT/RAISERROR sakinį, ir būtų kontrolės priklausomybė blokui $v_{if} \in P$, apimančiam IF...[ELSE]... sakinį, kad būtų tenkinama sąlyga: $v_{if} \in Dom(v_{rollback}) \wedge v_{rollback} \in Pdom(v_{if})$ $v_{if} \in Dom(v_{print/raiserror}) \wedge v_{print/raiserror} \in Pdom(v_{if}) \wedge v_{rollback} \in$ $Dom(v_{print/raiserror}) \wedge v_{print/raiserror} \in Pdom(v_{rollback})$

Pavadinimas	Verslo taisyklė <u>parodanti pranešimą</u> ir <u>draudžianti</u> duomenų modifikavimą prieštaraujantį tam tikrai sąlygai.
Programų sistemų lygmens taisyklės šablonas	JEI TENKINAMA <sąlyga>, NUTRAUKTI VEIKSMĄ IR PARODYTI PRANEŠIMĄ <„pranešimas“>
Verslo taisyklės šablonas	DUOMENŲ KOREGAVIMAS NEGALIMAS JEI <sąlyga> IR PARODOMAS PRANEŠIMAS: <„pranešimas“>
Pavyzdys	<pre> CREATE TRIGGER [dbo].[product_quantity_update] ON [dbo].[OrderItems] AFTER INSERT,UPDATE AS BEGIN SET NOCOUNT ON; declare @quantity int = null; SET @quantity = (select Quantity from inserted); IF (@quantity <= 0) BEGIN RAISERROR ('Klaida. Kiekis turi būti > 0.', 16, 1); ROLLBACK TRANSACTION END END </pre>

Šablonų aibė: (WHILE..) taisyklių tipas

Kontrolės srauto grafas:



Duomenų modifikavimas

VT šablonų pavyzdžiai tenkinantys CFG taisyklių tipą:

28 lentelė. Kontrolės srauto grafo (WHILE..) verslo taisyklės šablonas T_10

Pavadinimas	Verslo taisyklė <u>įrašanti</u> duomenis, jei duomenų modifikavimas tenkina tam tikrą sąlygą ir <u>tesia</u> sąlygos tikrinimą.
Apibūdinimas	Trigerio kūne turi būti while_statement sakiny, kuris savo bloke turi turėti insert sakinį. T.y. kol tam tikra sąlyga, trigeris įrašo apibrėžtus duomenis ir tęsia sąlygos tikrinimą.
Taisyklės formalus	Trigerio procedūros kontrolės srauto grafe CFG turi būti bent vienas kelias $P = \{v_i, \dots, v_n\}$, kuriame būtų blokas $v_{insert} \in P$, apimantis

Pavadinimas	Verslo taisyklė <u>įrašanti</u> duomenis, jei duomenų modifikavimas tenkina tam tikrą sąlygą ir <u>tesia</u> sąlygos tikrinimą.
aprašymas	<code>INSERT..[SET]..[FROM]..[WHERE]..</code> sakinį bei turintis kontrolės priklausomybę blokui $v_{while} \in P$, apimančiam <code>WHILE..</code> sakinį, kad būtų tenkinama sąlyga: $v_{while} \in Dom(v_{insert}) \wedge v_{insert} \in Pdom(v_{while})$
Programų sistemų lygmens taisyklės šablonas	KOL TENKINAMA <sąlyga>, ĮTERPTI DUOMENIS <faktas> IR TĘSTI
Verslo taisyklės šablonas	DUOMENŲ <faktas> ĮRAŠYMAS TĘSIASI TOL KOL <sąlyga>
Pavyzdys	<pre> ALTER TRIGGER [dbo].[product_settings_orders_update] ON [dbo].[ProductSettings] AFTER INSERT,UPDATE AS BEGIN SET NOCOUNT ON; declare @OrderID int = null; declare @OrderNR nchar(10) = '' DECLARE Orders_Cursor CURSOR FOR SELECT ID, OrderNr FROM Orders WHERE OrderState = 1; OPEN Orders_Cursor; FETCH NEXT FROM Orders_Cursor INTO @OrderID, @OrderNR; WHILE @@FETCH_STATUS = 0 BEGIN INSERT INTO [MokymaiDB].[dbo].[Messages] ([MessageType] , [MessageText]) VALUES (2 , 'Užsakymui Nr.: ' + @OrderNR + '. Reikia atspausdinti faktūra') FETCH NEXT FROM Orders_Cursor INTO @OrderID, @OrderNR; END; CLOSE Orders_Cursor; DEALLOCATE Orders_Cursor; END </pre>

29 lentelė. Kontrolės srauto grafo (WHILE..) verslo taisyklės šablonas T_11

Pavadinimas	Verslo taisyklė <u>atnaujinanti</u> duomenis, jei duomenų modifikavimas tenkina tam tikrą sąlygą ir <u>tesia</u> sąlygos tikrinimą.
Apibūdinimas	Trigerio kūne turi būti while_statement sakiny, kuris savo bloke turi turėti update sakinį. T.y. kol tam tikra sąlyga, triggeris atnaujiną apibrėžtus duomenis ir tęsia sąlygos tikrinimą.
Taisyklės formalus aprašymas	Trigerio procedūros kontrolės srauto grafe CFG turi būti bent vienas kelias $P = \{v_i, \dots, v_n\}$, kuriame būtų blokas $v_{update} \in P$, apimantis <code>UPDATE..[SET]..[FROM]..[WHERE]..</code> sakinį bei turintis kontrolės priklausomybę blokui $v_{while} \in P$, apimančiam <code>WHILE..</code> sakinį, kad būtų tenkinama sąlyga:

Pavadinimas	Verslo taisyklė <u>atnaujinanti</u> duomenis, jei duomenų modifikavimas tenkina tam tikrą sąlygą ir <u>tesia</u> sąlygos tikrinimą.
	$v_{while} \in Dom(v_{update}) \wedge v_{update} \in Pdom(v_{while})$
Programų sistemų lygmens taisyklės šablonas	KOL TENKINAMA <sąlyga>, ATNAUJINTI DUOMENIS <faktas> IR TĘSTI
Verslo taisyklės šablonas	DUOMENŲ <faktas> ATNAUJINIMAS TĘSIASI TOL KOL <sąlyga>
Pavyzdys	<pre> ALTER TRIGGER [dbo].[product_settings_price_update] ON [dbo].[ProductSettings] AFTER INSERT,UPDATE AS BEGIN SET NOCOUNT ON; WHILE (SELECT AVG(Price) FROM Products where Products.ID = (select ProductID from inserted)) < 10 BEGIN UPDATE Products SET Price = Price * (select PriceRate from inserted) WHERE ID = (select ProductID from inserted) END END </pre>

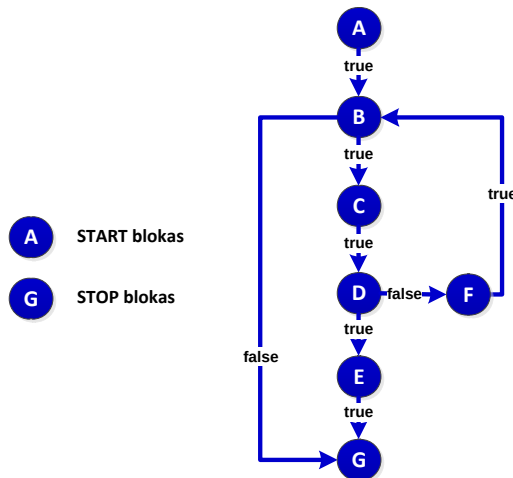
30 lentelė. Kontrolės srauto grafo (WHILE..) verslo taisyklės šablonas T_12

Pavadinimas	Verslo taisyklė <u>ištrinti</u> duomenis, jei duomenų modifikavimas tenkina tam tikrą sąlygą ir <u>tesia</u> sąlygos tikrinimą.
Apibūdinimas	Trigerio kūne turi būti while_statement sakiny, kuris savo bloke turi turėti delete sakinį. T.y. kol tam tikra sąlyga, trigeris ištrina apibrėžtus duomenis ir tęsia sąlygos tikrinimą.
Taisyklės formalus aprašymas	Trigerio procedūros kontrolės srauto grafe CFG turi būti bent vienas kelias $P = \{v_i, \dots, v_n\}$, kuriame būtų blokas $v_{delete} \in P$, apimantis DELETE..[SET]..[FROM]..[WHERE].. sakinį bei turintis kontrolės priklausomybę blokui $v_{while} \in P$, apimančiam WHILE.. sakinį, kad būtų tenkinama sąlyga: $v_{while} \in Dom(v_{delete}) \wedge v_{delete} \in Pdom(v_{while})$
Programų sistemų lygmens taisyklės šablonas	KOL TENKINAMA <sąlyga>, IŠTRINTI DUOMENIS <faktas> IR TĘSTI
Verslo taisyklės šablonas	DUOMENŲ <faktas> ŠALINIMAS TĘSIASI TOL KOL <sąlyga>
Pavyzdys	<pre> CREATE TRIGGER [dbo].[product_settings_price_update3] ON [dbo].[ProductSettings] AFTER INSERT,UPDATE AS BEGIN SET NOCOUNT ON; WHILE (SELECT AVG(Price) FROM Products) < (select Price from inserted) BEGIN DELETE Products </pre>

Pavadinimas	Verslo taisyklė <u>ištrinti</u> duomenis, jei duomenų modifikavimas tenkina tam tikrą sąlygą ir <u>tesia</u> sąlygos tikrinimą.
	<pre> WHERE Price < (select Price from inserted) END END </pre>

Šablonų aibė: (WHILE..IF..ELSE..) taisyklių tipas

Kontrolės srauto grafas:



Duomenų modifikavimas

VT šablonų pavyzdžiai tenkinantys CFG taisyklių tipą:

31 lentelė. Kontrolės srauto grafo (WHILE..IF..ELSE..) verslo taisyklės šablonas T_13

Pavadinimas	Verslo taisyklė <u>įrašanti</u> duomenis, jei duomenų modifikavimas tenkina tam tikrą sąlygą ir <u>tesia</u> sąlygos tikrinimą, išskyrus tenkinant kitą sąlygą, <u>nevykdo</u> duomenų modifikavimo.
Apibūdinimas	Trigerio kūne turi būti while_statement sakiny, kuris savo bloke turi turėti insert sakinį. Ir turi būti if_statement sakiny, kuris savo arba then_body bloke turi turėti brake sakinį ir else_body bloke turi turėti continue sakinį. T.y. kol tam tikra sąlyga, trigeris įrašo apibrėžtus duomenis ir tesia sąlygos tikrinimą, išskyrus, praleidžia duomenų modifikavimą, kai tenkinama kita sąlyga.
Taisyklės formalus aprašymas	Trigerio procedūros kontrolės srauto grafe CFG turi būti bent vienas kelias $P = \{v_i, \dots, v_n\}$ turintis kontrolės priklausomybę blokui $v_{while} \in P$, apimančiam WHILE.. sakinį, kuriame būtų: 1) blokas $v_{insert} \in P$, apimantis INSERT.. [SET] .. [FROM] .. [WHERE] .. sakinį, 2) blokas $v_{brake} \in P$, apimantis BRAKE sakinį, turintis kontrolės priklausomybę blokui $v_{if} \in P$, apimančiam IF... sakinį 3) blokas $v_{continue} \in P$, apimantis CONTINUE sakinį, turintis kontrolės

Pavadinimas	Verslo taisyklė <u>įrašanti</u> duomenis, jei duomenų modifikavimas tenkina tam tikrą sąlygą ir <u>tesia</u> sąlygos tikrinimą, išskyrus tenkinant kitą sąlygą, <u>nevykdo</u> duomenų modifikavimo.
	priklausomybę blokui $v_{else} \in P$, apimančiam ELSE... sakinį 4) kad būtų tenkinama sąlyga: $v_{while} \in Dom(v_{insert}) \wedge v_{insert} \in Pdom(v_{while}) \wedge$ $v_{insert} \in Dom(v_{if}) \wedge v_{if} \in Pdom(v_{insert}) \wedge$ $v_{if} \in Dom(v_{brake}) \wedge v_{brake} \in Pdom(v_{if}) \wedge$ $v_{continue} \in Dom(v_{else}) \wedge v_{else} \in Pdom(v_{continue})$
Programų sistemų lygmens taisyklės šablonas	KOL TENKINAMA <sąlyga>, ĮTERPTI DUOMENIS <faktas>. JEI TENKINAMA KITA <sąlyga>, TAI NEVYKDYTI VEIKSMO, KITU ATVEJU TĘSTI
Verslo taisyklės šablonas	DUOMENŲ <faktas> ĮRAŠYMAS TĘSIASI TOL KOL <sąlyga>, IŠSKYRUS KAI <sąlyga>.
Pavyzdys	<pre> CREATE TRIGGER [dbo].[product_settings_orders_update2] ON [dbo].[ProductSettings] AFTER INSERT, UPDATE AS BEGIN SET NOCOUNT ON; declare @OrderID int = null; declare @OrderNR nchar(10) = '' DECLARE Orders_Cursor CURSOR FOR SELECT ID, OrderNr FROM Orders WHERE OrderState = 1; OPEN Orders_Cursor; FETCH NEXT FROM Orders_Cursor INTO @OrderID, @OrderNR; WHILE @@FETCH_STATUS = 0 BEGIN INSERT INTO [MokymaiDB].[dbo].[Messages] ([MessageType] , [MessageText]) VALUES (2 , 'Užsakymui Nr.: ' + @OrderNR + '. Reikia atspausdinti faktūra') IF (GETDATE()) >= '2015-01-01' BREAK ELSE CONTINUE FETCH NEXT FROM Orders_Cursor INTO @OrderID, @OrderNR; END; CLOSE Orders_Cursor; DEALLOCATE Orders_Cursor; END </pre>

32 lentelė. Kontrolės srauto grafo (WHILE..IF..ELSE.) verslo taisyklės šablonas T_14

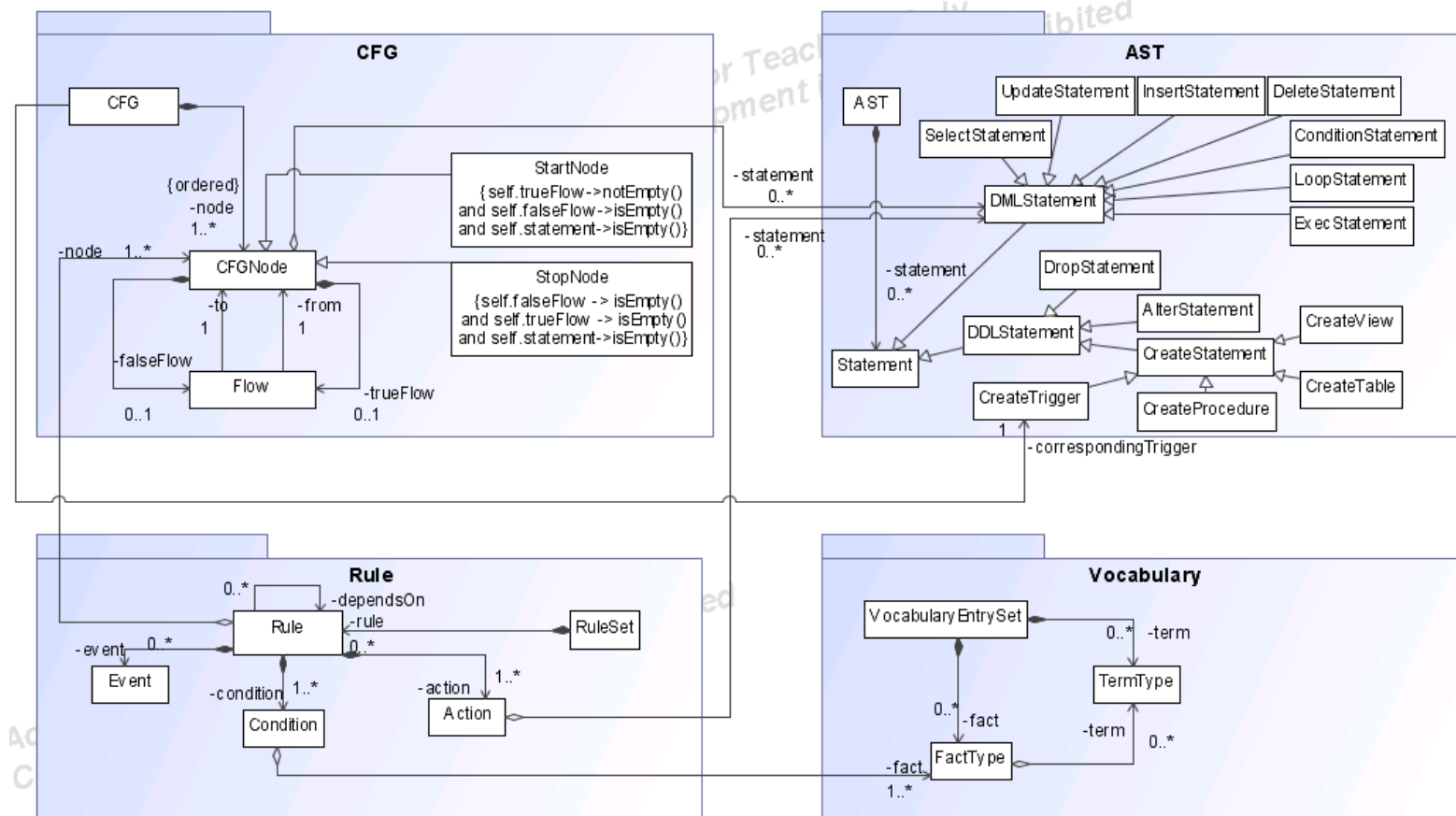
Pavadinimas	Verslo taisyklė <u>atnaujinanti</u> duomenis, jei duomenų modifikavimas tenkina tam tikrą sąlygą ir <u>tesia</u> sąlygos tikrinimą, išskyrus tenkinant kitą sąlygą, <u>nevykdo</u> duomenų modifikavimo.
Apibūdinimas	Trigerio kūne turi būti while_statement sakiny, kuris savo bloke turi turėti update sakinį.

Pavadinimas	Verslo taisyklė <u>atnaujinanti</u> duomenis, jei duomenų modifikavimas tenkina tam tikrą sąlygą ir <u>tesia</u> sąlygos tikrinimą, išskyrus tenkinant kitą sąlygą, <u>nevykdo</u> duomenų modifikavimo.
	Ir turi būti if_statement sakinys, kuris savo arba then_body bloke turi turėti brake sakinį ir else_body bloke turi turėti continue sakinį. T.y. kol tam tikra sąlyga, trigeris atnauja apibrėžtus duomenis ir tesia sąlygos tikrinimą, išskyrus, praleidžia duomenų modifikavimą, kai tenkinama kita sąlyga.
Taisyklės formalus aprašymas	Trigerio procedūros kontrolės srauto grafė <i>CFG</i> turi būti bent vienas kelias $P = \{v_i, \dots, v_n\}$ turintis kontrolės priklausomybę blokui $v_{while} \in P$, apimančiam WHILE...sakinį, kuriame būtų: 1) blokas $v_{insert} \in P$, apimantis UPDATE...[SET]...[FROM]...[WHERE]... sakinį, 2) blokas $v_{brake} \in P$, apimantis BRAKE sakinį, turintis kontrolės priklausomybę blokui $v_{if} \in P$, apimančiam IF... sakinį 3) blokas $v_{continue} \in P$, apimantis CONTINUE sakinį, turintis kontrolės priklausomybę blokui $v_{else} \in P$, apimančiam ELSE... sakinį 4) kad būtų tenkinama sąlyga: $v_{while} \in Dom(v_{update}) \wedge v_{update} \in Pdom(v_{while}) \wedge$ $v_{update} \in Dom(v_{if}) \wedge v_{if} \in Pdom(v_{update}) \wedge$ $v_{if} \in Dom(v_{brake}) \wedge v_{brake} \in Pdom(v_{if}) \wedge$ $v_{continue} \in Dom(v_{else}) \wedge v_{else} \in Pdom(v_{continue})$
Programų sistemų lygmens taisyklės šablonas	KOL TENKINAMA <sąlyga>, ATNAUJINTI DUOMENIS <faktas>. JEI TENKINAMA KITA <sąlyga>, TAI NEVYKDYTI VEIKSMO, KITU ATVEJU TĘSTI
Verslo taisyklės šablonas	DUOMENŲ <faktas> ATNAUJINIMAS TĘSIASI TOL KOL <sąlyga>, IŠSKYRUS KAI <sąlyga>.
Pavyzdys	<pre> CREATE TRIGGER [dbo].[product_settings_price_update2] ON [dbo].[ProductSettings] AFTER INSERT,UPDATE AS BEGIN SET NOCOUNT ON; declare @actionID int = null; WHILE (SELECT AVG(Price) FROM Products) < 10 BEGIN UPDATE Products SET Price = Price * (select PriceRate from inserted) IF (SELECT MAX(Price) FROM Products) > 50 BREAK ELSE CONTINUE END END </pre>

33 lentelė. Kontrolės srauto grafo (WHILE..IF..ELSE.) verslo taisyklės šablonas T_15

Pavadinimas	Verslo taisyklė <u>ištrinantį</u> duomenis, jei duomenų modifikavimas tenkina tam tikrą sąlygą ir <u>tesia</u> sąlygos tikrinimą, išskyrus tenkinant kitą sąlygą, <u>nevykdo</u> duomenų modifikavimo.
Apibūdinimas	Trigerio kūne turi būti while_statement sakiny, kuris savo bloke turi turėti delete sakinį. Ir turi būti if_statement sakiny, kuris savo arba then_body bloke turi turėti brake sakinį ir else_body bloke turi turėti continue sakinį. T.y. kol tam tikra sąlyga, trigeris ištrina apibrėžtus duomenis ir tęsia sąlygos tikrinimą, išskyrus, praleidžia duomenų modifikavimą, kai tenkinama kita sąlyga.
Taisyklės formalus aprašymas	Trigerio procedūros kontrolės srauto grafe CFG turi būti bent vienas kelias $P = \{v_i, \dots, v_n\}$ turintis kontrolės priklausomybę blokui $v_{while} \in P$, apimančiam WHILE.. sakinį, kuriame būtų: 1) blokas $v_{insert} \in P$, apimantis DELETE.. [SET] .. [FROM] .. [WHERE] .. sakinį, 2) blokas $v_{brake} \in P$, apimantis BRAKE sakinį, turintis kontrolės priklausomybę blokui $v_{if} \in P$, apimančiam IF... sakinį 3) blokas $v_{continue} \in P$, apimantis CONTINUE sakinį, turintis kontrolės priklausomybę blokui $v_{else} \in P$, apimančiam ELSE... sakinį 4) kad būtų tenkinama sąlyga: $v_{while} \in Dom(v_{delete}) \wedge v_{delete} \in Pdom(v_{while}) \wedge$ $v_{delete} \in Dom(v_{if}) \wedge v_{if} \in Pdom(v_{delete}) \wedge$ $v_{if} \in Dom(v_{brake}) \wedge v_{brake} \in Pdom(v_{if}) \wedge$ $v_{continue} \in Dom(v_{else}) \wedge v_{else} \in Pdom(v_{continue})$
Programų sistemų lygmens taisyklės šablonas	KOL TENKINAMA <sąlyga>, IŠTRINTI DUOMENIS <faktas>. JEI TENKINAMA KITA <sąlyga>, TAI NEVYKDYTI VEIKSMO, KITU ATVEJU TĘSTI
Verslo taisyklės šablonas	DUOMENŲ <faktas> ŠALINIMAS TĘSIASI TOL KOL <sąlyga>, IŠSKYRUS KAI <sąlyga>.
Pavyzdys	<pre> CREATE TRIGGER [dbo].[product_settings_price_update3] ON [dbo].[ProductSettings] AFTER INSERT,UPDATE AS BEGIN SET NOCOUNT ON; declare @actionID int = null; WHILE (SELECT AVG(Price) FROM Products) < (select Price from inserted) BEGIN DELETE Products WHERE Price < (select Price from inserted) IF (SELECT MAX(Price) FROM Products) < 50 BREAK ELSE CONTINUE END END </pre>

AST, CFG, VŽ ir VT šablonų tipų metamodelių susiejimas



14 pav. AST, CFG, VŽ ir VT šablonų tipų metamodelių susiejimas

C priedas.

Verslo taisyklės šablono pritaikymo algoritmo pseudokodas

Input:

CFG[] CFG; \visų trigerių suformuotų CFG aibė

Node[] visited; \aplankytų blokų aibė

Template[] templates; \šablonų aibė

enum RuleType{IfStatement, WhileStatement}; \šablonų taisyklių tipai

Output:

RuleSet[] ruleSet; \taisykliu aibe - kandidatai

```
Main(){
    foreach(cfg in CFG){
        visited[] <- empty \tuščią aplankytų blokų aibę
        AnalyseCFG(cfg.node.Get(typeof StartNode)); \pradedama nuo
        pradinio bloko
    }
}

//čia visas patikrinimas turetų eiti:
if(node.statements.Contains(typeof IfStatement)){
    //pritaikomi IF sakinių šablonai
    MatchForRule(node, RuleType.IfStatement);
    GetRules(node.trueFlow);
    GetRules(node.falseFlow);
}
else if(node.statements.Contains(typeof WhileStatement)){
    //pritaikomi WHILE sakinių šablonai
    MatchForRule(node, RuleType.WhileStatement);
    GetRules(node.trueFlow);
    GetRules(node.falseFlow);

    //else if (blokas turi kitą suprantamą sakinį <- galimybė papildyti
    tikrinamų taisyklių tipų aibę)
    //kviesti MatchForRule(...)
} else if(!(node typeof StopNode)) {
    //kadangi tai ne salygos (IfStatement, WhileStatement) sakiny
    //tai jis turi tik trueFlow
    GetRules(node.trueFlow);
}

}

}

}
```

```

MatchForRule(Node node, RuleType type){
    switch (type){
        case RuleType.IfStatement:
            foreach(Template template in templates.Get(typeof
IfStatement)){ //atrenkamas sablonas kuris atitinka tipa
                //patikrinam atitikima betn vienam sablonui
                //cia yra tikrinimas atitikimo konkreiam sablonui
atsizvelgiant i formaliai aprasytus tipus
                //jei taisykle yra turi "tevine", nuo kurios priklauso
taisykle

                //tai pridedam tame kode CheckForPattern
                Rule rule = CheckForPattern(node, template);

                if (rule != null) {
                    ruleSet.Add(rule);
                }
            }
            break;
        case RuleType.WhileStatement:
            foreach(Template template in templates.Get(typeof
WhileStatement)){//atrenkamas sablonas kuris atitinka tipa
                //patikrinam atitikima bet vienam sablonui
                Rule rule = CheckForPattern(node, template); //cia yra
tikrinimas atitikimo konkreiam sablonui atsizvelgiant i formaliai aprasytus tipus
                if (rule != null) {
                    ruleSet.Add(rule);
                }
            }
            break;
    }
}

Rule CheckForPattern(Node node, Template template){
    if(node atitinka template){
        Rule rule <- sukuriam nauja taisykle
        Event[] event <- sukuriam nauja ivyki ir uzpildom ji is informacijos apie
trigeriu ivykius: node.GetOwner().correspondingTrigger.Event
        Action[] action <- istraukiam to node.trueFlow bloka
        Condition[] condition <- istraukiam SQL israiskas, kurias tikrinsim per
Vocabulary uzpildyma
        return rule;
    }else{
        return null;
    }
}

```